Object Model

Click on an object to view the description.
Modified Event

Triggered when an object or collection in the drawing has been modified.

See Also | Example

Signature

object.Modified()

Object

All Land objects can trigger the Modified Event An object expression that evaluates to a valid container object.

Remarks

This event will be triggered whenever the object is modified. Modification includes whenever the value of a property is set, even if the new value is equal to the current value.

When coding in VBA, you must provide an event handler for all objects enabled for the Modified event. If you do not provide a handler, VBA may terminate unexpectedly.

No events will be fired while a modal dialog is being displayed.

Refer to the Land ActiveX and VBA Developer's Guide for a list of support events.
Methods

A

Add Creates a member object and adds it to the appropriate collection.

AddCurve Appends a curve to the alignment or parcel.

AddLabelAt Adds a contour label at the specified location.

AddLine Appends a line to a parcel.

AddSpiral Appends a spiral to the alignment.

AddTangent Appends a tangent to the alignment.

AddToAllElevations Adds a fixed amount to all elevations in the Surface.

AlignmentFromObjectID Given the object ID for an entity, returns the Alignment name.

AreaVolume Given an array of points, returns the cut, fill, and net volume.

ArrayToPointString Given an array of point numbers, returns a string with all of the point numbers.

B

Build The Build method will recalculate the Surface.

C

ClearOverides Clears all overrides for the project database point group.

Composite Creates a composite surface from two existing surfaces.

ConvertToCurrentAreaDisplay Converts an area value using the LinearUnit and AreaDisplayUnit settings.
**ConvertToCurrentVolumeDisplay** Converts a volume value using the LinearUnit and VolumeDisplayUnit settings.

**Copy** Copies a Prototype or Surface.

**Delete** Deletes a specified object.

**DifferenceGrid** Creates a surface from two existing surfaces using the Grid method.

**DoubleToStaFormat** Given a double, returns a string with the Station Format applied.

**EastNorthToXy** Given an Easting and Northing, will return the AutoCAD XY

**ElevationAt** (civil engineering feature) Given a station, returns the elevation for a profile.

**ExternalStaToInternal** Given an external Station value (one using Station Equations), returns the internal Station value.

**FindAllConnectingEdges** Returns all edges for a given Surface point.

**FindAllFaces** Redefines the collection with all faces on the surface.

**FindConnectingEdge** Given a Surface point, retrieves the closest edge.

**FindFace** Given an Easting / Northing coordinate, returns the Face.

**FindPath** Give two coordinates, reads in all of the Faces that lie on that line.

**FindPoint** Given an Easting / Northing coordinate, finds the closest Surface.
G

**GetBoundingBox** Gets the bounding box for the surface.

**GetDouble** Gets the specified preference setting.

**GetElevation** Given a Northing and Easting, return the elevation.

**GetInteger** Gets the specified preference setting.

**GetLayerName** *(civil engineering feature)* Returns the layer names of the alignment cross section or alignment profile.

**GetStaStrWithEquations** Given an internal station value, returns that station as a formatted string with Station Equations applied.

**GetStaWithEquations** Given an internal station value, returns that station value with Station Equations applied.

**GetString** Gets the specified preference setting.

H

I

**Import** Draws the alignment, cross section, existing ground profile, parcel, or surface into the current drawing.

**InstantGrade** *(civil engineering feature)* Given a station, returns the instantaneous grade and the algebraic difference.

**Item** Gets the member object at a given index in a collection.

J

K

L

**LineIntersection** Returns the intersection points of a line with the Alignment.
LoadSetupProfile  Load the specified drawing setup profile.

LockPoints  Locks a list of point numbers.

M

N

NewProjectBased  Creates a new AutoCAD Land Desktop document.

O

OffsetElevationToXy  (civil engineering feature) Given an offset and elevation, return the AutoCAD XY coordinates.

Open  Opens an existing AutoCAD Land Desktop drawing or surface.

OpenProjectBased  Opens an existing AutoCAD Land Desktop drawing file (DWG).

P

Paste  The Paste method will take another Surface and paste it onto the given Surface.

PerpIntersection  Returns the perpendicular intersection of a point with an Alignment.

PointByNumber  Returns a point in the project database by number.

PointCodeDescription  (civil engineering feature) Returns the description of a cross section point code.

PointLocation  Given a station and offset, returns the Northings and Eastings.

PointNumberFromObjID  Given a AutoCAD object ID, PointNumberFromObjID will return the point number.

PointStringToArray  Given a point string, returns an array with all of the point numbers.
**ProfileByType** (civil engineering feature) Returns the Existing or Finished Ground Profile by type.

**Q**

**R**

**RemoveAll** Remove all method for Alignments and StationEquations.

**RemoveAllLabels** Removes all contour labels.

**RemoveLabelAt** Removes the contour label closest to the specified location.

**Rename** Renames a surface.

**S**

**SampleElevations** Given a start and end point, returns an array of derived points from the Surface.

**Save** Writes objects to specific databases.

**SaveAsDefault** Saves the DatabasePreferences to the registry.

**SaveSetupProfile** Saves the current drawing settings to a drawing setup profile.

**SectionByStation** (civil engineering feature) Returns a Cross Section object by specifying its station.

**SectionVolume** (civil engineering feature) Calculates the Cross Section volumes.

**SetBoundingBox** Sets the bounding box for the surface.

**SetDouble** Sets the specified preference setting.

**SetInteger** Sets the specified preference setting.

**SetReferenceCurve** Sets the reference curve for the Curve Text.
**SetString** Sets the specified preference setting.

**StationElevationToXy** (civil engineering feature) Given an alignment cross section or alignment profile station and elevation, return the AutoCAD XY coordinates.

**StationOffset** Given a Northings and Eastings, returns the station, and an offset, and the polar direction.

T

U

**UnlockPoints** Unlocks a list of point numbers.

V

W

X

**XyToEastNorth** Given a AutoCAD X, Y, returns the Easting and Northing.

**XyToOffsetElevation** (civil engineering feature) Given an AutoCAD XY coordinates, return the cross section offset and elevation.

**XyToStationElevation** (civil engineering feature) Given an AutoCAD XY coordinates, return the profile station and elevation.

Y

Z
Add Method

Creates a member object and adds it to the appropriate collection.

See Also | Example

Signature: Overview

- Alignments
- Boundaries
- BreakLines
- CogoPoints
- ContourItems
- DEMFiles
- DescriptionKeyFile
- DescriptionKeyFiles
- EGProfiles (Civil Engineering Feature)
- FGProfiles (Civil Engineering Feature)
- Parcels
- PointFiles
- PointGroupNames
Creates a new Alignment and adds it to the Alignments collection.

**Signature**

RetVal = object.Add(Name, StartingStation)

object

**Alignments** The object or objects this property applies to.

Name

String; input-only
New alignment name.

StartingStation

Double; input-only
The starting station.

RetVal

**Alignment** object
The newly added Alignment object.

**Remarks**

The new alignment must be saved for any changes to take affect. You can add a new alignment to the collection and then define the underlying geometry. Be advised that if you save an alignment without later adding any tangents, curves or spirals, you will not be able to edit the geometry using the alignment editor.
Adds a Boundary to the Surface.

**Signature**

RetVal = object.Add(Type, bIsBreakLine, Coordinates[, Description])

- **object**
  - Boundaries
  - The object or objects this property applies to.

- **Type**
  - eAeccBoundaryType enum; input-only
    - kBoundaryTypeShow: Boundary is visible
    - kBoundaryTypeHide: Boundary is hidden
    - kBoundaryTypeOuter: Boundary is the outer most boundary
    - kBoundaryTypeUninit: Boundary in not initialized.

- **bIsBreakLine**
  - Boolean; input only
    - TRUE: Boundary is a breakline
    - FALSE: Boundary is not a breakline

- **Coordinates**
  - Variant (array of doubles); input-only
    - The points in Easting, Northing, Elevation format.

- **Description**
  - String; input only; optional
    - The description of the Boundary.
**Boundary object**
The newly added Boundary object.

**Remarks**
Outer boundaries are always visible and must be the first boundary defined.

Creates a new BreakLine for the Surface.

**Signature**
RetVal = object.Add(Coordinates[, Description])

object

**BreakLines**
The object or objects this property applies to.

Coordinates

Variant (array of doubles); input-only
The points in Easting, Northing, Elevation format.

Description

String; input-only; optional
The description for the BreakLine.

RetVal

**BreakLine object**
The newly added BreakLine object.

**Remarks**
Add creates the BreakLine by adding it to the Surface. It will also add the BreakLine to the **BreakLines** collection. Only a Standard BreakLine can be added. The Surface must be built for the change to take affect.

Creates a new CogoPoint and adds it to the collection for the project
database.

**Signature**

RetVal = object.Add(Coordinates, Format)

**object**

*CogoPoints*
The object or objects this property applies to.

**Coordinates**

Variant (2 or 3 element array of doubles); input-only. If the array has 2 elements, no elevation will be assigned.

Double[0]: Easting, or X based on PointStyle

Double[1]: Northing, or Y based on PointStyle

Double[2]: Optional: Elevation or kPointNoElevation

**Format**

eAeccCoordinateFormat enum; input-only

kCoordinateFormatENZ: Point is in Easting, Northing, elevation

kCoordinateFormatXYZ: Point is in X, Y, elevation. AutoCAD coordinates

**RetVal**

*CogoPoint* object
The newly added *CogoPoint* object.
Adds a ContourItem to the Surface definition.

**Signature**

RetVal = object.Add(Coordinates)

object

ContourItems
The object or objects this property applies to.

Coordinates

Variant (array of doubles); input-only
An array of doubles representing Easting, Northing, Elevation.

RetVal

ContourItem object
The newly added ContourItem object.

---

Adds a Digital Elevation Model file to the Surface definition.

**Signature**

RetVal = object.Add(FullName, [CoordinateZone])

object

DEMFiles
The object or objects this property applies to.

FullName

String; input-only
Full name of the DEM file to be added.

CoordinateZone

String; input-only
The coordinate system syntax for the zone.

RetVal
**DEMFFile object**
The newly added **DEMFFile** object.

---

Creates a new DescriptionKey.

**Signature**

RetVal = object.Add(syntax)

object

**DescriptionKeyFile**
The object or objects this property applies to.

syntax

String; input-only
The syntax of the new **DescriptionKey**

RetVal

**DescriptionKey** object
The newly added **DescriptionKey** object.

---

Creates a new DescriptionKeyFiles and adds it to the collection for the project database.

**Signature**

RetVal = object.Add(Name)

object

**DescriptionKeyFiles**
The object or objects this property applies to.

Name

String; input-only
The name of the new **DescriptionKeyFiles**

RetVal
The newly added **DescriptionKeyFile** object.

Creates a new EGProfile and adds it to the EGProfiles collection.

**Signature**

RetVal = Object.Add(Type, SurfaceName)

**object**

**EGProfiles**

The object or objects this property applies to.

**Type**

eAeccEGProfilesType enum; input-only

The type for the new EGProfile.

kEgCenter: Existing ground center profile

kEgLeft: Existing ground left profile

kEgNone: Existing ground none

kEgRight: Existing ground right profile

**String**

String; input-only

The name of the surface.

**RetVal**

EGProfile object

The newly added EGProfile object.

---

Adds a Finished Grade Profile to the project
**Signature**

RetVal = Object.Add(Type)

object

**FGProfiles**
The object or objects this property applies to.

**Type**

eAeccFGProfileType enum; input-only
The type of finished grade profile to add.

kFgCenter: Finished grade center profile
kFgDitchLeft: Finished grade left ditch profile
kFgDitchRight: Finished grade right ditch profile
kFgLeft1: Finished grade left 1 profile
kFgLeft2: Finished grade left 2 profile
kFgLeft3: Finished grade left 3 profile
kFgLeft4: Finished grade left 4 profile
kFgLeft5: Finished grade left 5 profile
kFgLeft6: Finished grade left 6 profile
kFgLeft7: Finished grade left 7 profile
kFgLeft8: Finished grade left 8 profile
kFgNone: Finished grade none
RetVal

**FGProfile** object
The newly added **FGProfile** object.

**Remarks**

A Finished Grade Profile can not start or end with a Vertical Curve.

---

Creates a new Parcel and adds it to the ExistingParcelAligns collection.

**Signature**

RetVal = Object.Add(Name)

object

**Parcels**
The object or objects this property applies to.

Name

String; input-only
The name of the parcel.

RetVal

Parcel object
The newly added Parcels object.

Adds a point file to the PointFiles collection.

**Signature**

RetVal = object.Add(FullName)

object

PointFiles
The object or objects this property applies to.

FullName
String; input-only
Full name of the point file to be added.

RetVal

PointFile object
The newly added PointFile object.

**Remarks**

PointFile.Add does not check for the existence of the file. The new PointFile will use the default file format of Point Number, Easting, Northing, Elevation (space-delimited). Each field in the point file must be separated with a single space.

---

Adds a new PointGroupName to the PointGroupNames collection.

**Signature**

RetVal = object.Add(Name)

object
**PointGroupNames**
The object or objects this property applies to.

Name
String; input-only
Name of the PointGroupName to add.

RetVal
**PointGroupName** object
The newly added **PointGroupName** object.

Add a new point group to the project database.

**Signature**
RetVal = object.Add(Name, PointList)

object
**PointGroups**
The object or objects this property applies to.

GroupName
String; input-only
The new point group name.

PointList
String; input-only
The point list.

RetVal
**PointGroup** object
The newly added **PointGroup** object.

Creates a new project.

**Signature**
RetVal = object.Add(ProjectName, PrototypeName[, CreatePDb])

object

Projects
The object or objects this property applies to.

ProjectName
String; input-only
The name of the project to delete.

PrototypeName
String; input-only
The name of the prototype to use for the project.

CreatePDb
Boolean; input-only; optional
True: The Cogo Point database will be created (default).
False: The Cogo Point database will not be created

RetVal

Project object
The newly added Project object.

Remarks
The new project is created at the ProjectPath directory. The default subfolder to save drawings in is the \dwg subfolder.

If CreatePDb is True, the Cogo Point database will be created. The Description field size will be 32 characters. Point Names will be enable with a size of 16 characters. The default value for CreatePDb is True.

----------------------------------------------------------------------------------------------------------------------------

Adds a PVI.

Signiture

RetVal = Object.Add(Station, Elevation, Curvelength)
object **PVIs**
The object or objects this property applies to.

**Station**

Double; input-only
The Station for the new PVI.

**Elevation**

Double; input-only
The Elevation for the new PVI.

**CurveLength**

Double; input-only
The CurveLength for the new PVI.

**RetVal**

**PVI** object
The newly added **PVI** object.

---

Adds a station equation into the alignment.

**Signature**

RetVal = object.Add(StationBack, StationAhead, Type)

object **StationEquations**
The object or objects this property applies to.

**StationBack**

Double; input-only
The back station for the station equation.

**StationAhead**

Double; input-only
The ahead station for the station equation.
Type

eAeccStationEquationType enum; input-only

kIncreasing: Station numbers increase along the alignment

kDecreasing: Station numbers decrease along the alignment

RetVal

StationEquation object
The newly added StationEquation object.

Creates a new Surface and adds it to the Surfaces collection.

Signature

RetVal object.Add(SurfaceName)

object

Surfaces
The object or objects this property applies to.

SurfaceName

String; input-only
New surface name.

RetVal

Surface object
The newly added Surface object.

Remarks

If the name of the surface is blank, then a temporary surface will be created. A temporary surface is useful for generating a surface in an AutoCAD drawing. A temporary surface can not be saved.
AddCurve Method

Appends a curve to the alignment or parcel.

See Also | Example

Signature: Overview

1 Alignment

1 Parcel

Appends a curve to the alignment.

Signature

RetVal = object.AddCurve(StartEasting, StartNorthing, CenterEasting, CenterNorthing, EndEasting, EndNorthing, bCCWFlag)

object

Alignment The object or objects this property applies to.

StartEasting

Double; input-only
The Easting coordinate for the beginning of the curve.

StartNorthing

Double; input-only
The Northing coordinate for the beginning of the
curve.

CenterEasting
Double; input-only
The Easting coordinate for the center of the curve.

CenterNorthing
Double; input-only
The Northing coordinate for the center of the curve.

EndEasting
Double; input-only
The Easting coordinate for the end of the curve.

EndNorthing
Double; input-only
The Northing coordinate for the end of the curve.

bCCWFlag
Boolean; input-only
TRUE: If the curve is drawing in a counterclockwise direction.
FALSE: If the curve is drawing in a clockwise direction.

RetVal
\texttt{AlignCurve} object
The Newly added AlignCurve object

**Remarks**

A check will be made to see if the beginning of the new curve is equal to the end of the previous entity. If the two are not equal, an error (AECC_E_ALN_INVALID_ENDPOINT) will be generated and the curve will not be added.

No check will be made to see if the new curve is tangent to the previous entity.
Appends a curve to the parcel.

**Signature**

RetVal = object.AddCurve(StartEasting, StartNorthing, CenterEasting, CenterNorthing, EndEasting, EndNorthing, bCCWFlag)

**object**

*Parcel*

The object or objects this property applies to.

**StartEasting**

Double; input-only

The Easting coordinate for the beginning of the curve.

**StartNorthing**

Double; input-only

The Northing coordinate for the beginning of the curve.

**CenterEasting**

Double; input-only

The Easting coordinate for the center of the curve.

**CenterNorthing**

Double; input-only

The Northing coordinate for the center of the curve.

**EndEasting**

Double; input-only

The Easting coordinate for the end of the curve.

**EndNorthing**

Double; input-only

The Northing coordinate for the end of the curve.

**bCCWFlag**

Boolean; input-only
TRUE: If the curve is drawing in a counterclockwise direction.
FALSE: If the curve is drawing in a clockwise direction.

RetVal

ParcelCurve object
The Newly added ParcelCurve object.
AddLabelAt Method

Adds a contour label at the specified location.

See Also | Example

Signature

object.AddLabelAt(X, Y)

object

AeccContour The object or objects this property applies to.

X

Double; input-only
The X coordinate of the new label.

Y

Double; input-only
The Y coordinate of the new label.

Remarks

The label will be placed at the closest location on the contour based on the X, Y point.
**AddLine Method**

Appends a line to the parcel.

**Signature**

RetVal = object.AddParcel(StartEasting, StartNorthing, EndEasting, EndNorthing)

- **object**
  - *Parcel* The object or objects this property applies to.

- **StartEasting**
  - Double; input-only
  - The Easting coordinate for the beginning of the line.

- **StartNorthing**
  - Double; input-only
  - The Northing coordinate for the beginning of the line.

- **EndEasting**
  - Double; input-only
  - The Easting coordinate for the end of the line.

- **EndNorthing**
  - Double; input-only
  - The Northing coordinate for the end of the line.

- **RetVal**
ParcelLine object
The newly added ParcelLine object.
AddSpiral Method

Appends a spiral to the alignment.

**Signature**

RetVal = object.AddSpiral(StartEasting, StartNorthing, PiEasting, PiNorthing, EndEasting, EndNorthing, LExt, LOffset, SpiralType1)

object

Alignment The object or objects this property applies to.

StartEasting

Double; input-only
The Easting coordinate for the beginning of the spiral.

StartNorthing

Double; input-only
The Northing coordinate for the beginning of the spiral.

PiEasting

Double; input-only
The Easting coordinate for the PI of the spiral.

PiNorthing

Double; input-only
The Northing coordinate for the PI of the spiral.

**EndEasting**

Double; input-only
The Easting coordinate for the end of the spiral.

**EndNorthing**

Double; input-only
The Northing coordinate for the end of the spiral.

**LExt**

Double; input-only
The external length of the spiral. 0.0 for a simple spiral.

**LOffset**

Double; input-only
The offset distance. 0.0 for non-offset spirals.

**SpiralType1**

eAeccSpiralType enum; input-only
- **kClothoid:** Clothoid spiral
- **kSinusoid:** Sinusoid spiral
- **kCosinusoid:** Cosinusoid spiral
- **kQuadratic:** Quadratic spiral

**RetVal**

**AlignSpiral** object
The newly added AlignSpiral object.

**Remarks**

A check will be made to see if the beginning of the new spiral is equal to
the end of the previous entity. If the two are not equal, an error (AECC_E_ALN_INVALID_ENDPOINT) will be generated and the spiral will not be added.

No check will be made to see if the new spiral is tangent to the previous entity.

Lext represents the length from the TS to the External Point in a compound spiral condition.
AddTangent Method

Appends a tangent to the alignment.

See Also | Example

Signature

RetVal = object.AddTangent(StartEasting, StartNorthing, EndEasting, EndNorthing)

object

Alignment The object or objects this property applies to.

StartEasting

Double; input-only
The Easting coordinate for the beginning of the tangent.

StartNorthing

Double; input-only
The Northing coordinate for the beginning of the tangent.

EndEasting

Double; input-only
The Easting coordinate for the end of the tangent.

EndNorthing

Double; input-only
The Northing coordinate for the end of the tangent.
RetVal

**AlignTangent** object
The newly added AlignTangent object.

**Remarks**

A check will be made to see if the beginning of the new tangent is equal to the end of the previous entity. If the two are not equal, an error (AECC_E_ALN_INVALID_ENDPOINT) will be generated and the tangent will not be added.
AddToAllElevations Method

Adds a fixed amount to all elevations in the Surface.

See Also | Example

Signature

object.AddToAllElevations(Delta)

object

Surface The object or objects this property applies to.

Delta

Double; input-only
Elevation to be added / subtracted to all elevations.

Remarks

AddToAllElevations effectively raises or lowers the surface by DeltaElevation. The Surface will not be rebuilt but will be saved when AddToAllElevations is called.

Edit history must be turned on when the Surface is rebuilt for the changes to been seen.
AlignmentFromObjectID Method

Given the object ID for an entity, returns the Alignment name.

See Also | Example

Signature

RetVal = object.AlignmentFromObjectID(ObjectID)

object

Alignments The object this property applies to.

ObjectID

Long; read-only
The Object ID for the entity.

RetVal

Variant (array of strings)
An array of Alignment names.
AreaVolume Method

Given an array of points defining an area, returns the cut, fill, and net volume.

See Also | Example

Signature

object.AreaVolume(Tolerance, Coordinates, Cut, Fill, Net)

object

Surface The object or objects this property applies to.

Tolerance

Double; input-only

Coordinates

Variant (array of doubles); input-only
A closed array of 3D points defining an area

Cut

Double; output-only
The cut volume.

Fill

Double; output-only
The fill volume.

Net

Double; output-only
The net volume.
ArrayToPointString Method

Given an array of point numbers, returns a string with all of the point numbers.

See Also | Example

Signature

RetVal = object.PointStringToArray(Points)

object

CogoPoints The object or objects this property applies to.

Points

Variant (array of longs); input-only
Each element is a point number.

RetVal

String
Comma delimited string of points to be locked with groups separated by hyphens.

Remarks

The returned point string will be condensed; if the array has point 1 through 10, then the point string returned is "1-10".
Build Method

Builds Watersheds or recalculates the Surface.

See Also | Example

Signature: Overview

Surface

WaterSheds

The Build method will recalculate the Surface.

Signature

object.Build()

object

Surface The object or objects this property applies to.

Remarks

When changes are made to a Surface, the Surface needs to be recalculated. The Build method will do the recalculation. The status of a surface will only be accurate when the time between a Build and an addition of Surface Input data is greater than the resolution of the file system.

Builds the WaterSheds for the Surface.
**Signature**

object.Build(MinDepressionDepth, MinDepressionArea, bExceedBoth)

object

**WaterSheds**
The object or objects this property applies to.

MinDepressionDepth

Double; input-only
The minimum depression depth for each watershed.

MinDepressionArea

Double; input-only
The minimum depression area for each watershed.

bExceedBoth

Boolean; input-only
TRUE: Both minimum depression conditions must be met.
FALSE: Either minimum depression setting must be met.

**Remarks**

Any change in the Surface statistics will not occur until the Surface and Surfaces objects are released and created again. For example, WaterSheds.Build may add TinPoints to the Surface. The **NumberOfPoints** for the Surface will not be updated until both the Surfaces collection and the Surface object are updated.
ClearOverrides Method

Clears all overrides for the project database point group.

**Signature**

object.Clear Overrides()

object

PointGroup The object or objects this property applies to.
Composite Method

Creates a composite surface from two existing surfaces.

See Also | Example

Signature

RetVal = object.Composite(NewSurface, ParentSurface, ChildSurface)

object

Surfaces The object or objects this property applies to.

NewSurface

String; input-only
The name of the new surface to be created.

ParentSurface

String; input-only
The name of the parent surface.

ChildSurface

String; input-only
The name of the child surface.

RetVal

Surface object
The newly added Surface object.
ConvertToCurrentAreaDisplay Method

Converts an area value using the LinearUnit and AreaDisplayUnit settings.

See Also | Example

**Signature**

RetVal = object.ConvertToCurrentAreaDisplay(Area)

object

[DatabasePreferences](#) The object or objects this property applies to.

Area

Double; input-only
The area to convert.

RetVal

Double
The resulting area.
ConvertToCurrentVolumeDisplay Method

Converts a volume value using the LinearUnit and VolumeDisplayUnit settings.

See Also | Example

Signature

RetVal = object.ConvertToCurrentVolumeDisplay(Volume)

object

DatabasePreferences The object or objects this property applies to.

Volume

Double; input-only
The volume to convert.

RetVal

Double
The resulting volume.
Copy Method

Creates a copy of an existing object.

See Also | Example

Signature: Overview

I Prototypes

I Surfaces

Copies a prototype.

Signature

RetVal = object.Copy(FromName, ToName)

object

Prototypes The object or objects this property applies to.

FromString

String; input-only
The name of the prototype you wish to copy.

ToName

String; input-only
The name of the copy.

RetVal
**Prototype object**
The newly added Prototype object.

**Remarks**
The new prototype is created at the [PrototypePath](#) directory.

---

Copies a surface.

**Signature**

RetVal = object.Copy(SurfaceName)

[object](#)

[**Surfaces**](#)
The object or objects this property applies to.

SurfaceName

String; input-only
The name of the surface to be copied.

RetVal

[**Surface**](#) object
The newly added Surface object.

**Remarks**
The new surface created will have "Copy of" prefixed to the new surface name.
Delete Method

Delete a specified object.

Signature: Overview

- Alignments
- Boundaries
- BreakLines
- CogoPoints
- ContourItems
- DEMFiles
- DescriptionKeyFile
- DescriptionKeyFiles
- EGProfiles (Civil Engineering Feature)
- FGProfiles (Civil Engineering Feature)
- Parcels
- PointFiles
- PointGroupNames
 Deletes an Alignment and removes it from the Alignments collection.

**Signature**

object.Delete(Name)

object  

---

**Alignments** The object or objects this property applies to.

Name  

String; input-only  
The alignment name to delete.

**Remarks**

Delete will remove the alignment from the database and the drawing. If the deleted alignment was set current there will not be any alignment set current after the delete has taken place.

---

Given a Boundary identifier, deletes that Boundary from the Surface.

**Signature**

object.Delete(Id)
object

**Boundaries**
The object or objects this property applies to.

Id

Long; input-only
The identifier of the boundary.

Delete removes the BreakLine from the Surface. It will also remove the BreakLine from the **BreakLines** collection. The Surface must be built for the change to take affect.

---

Deletes the point from the project database.

**Signature**

object.Delete(PointNumber)

---

object

**CogoPoints**
The object or objects this property applies to.
PointNumber

Long; input-only
The Cogo Point number to be deleted.

Deletes a Digital Elevation Model file from the Surface.

**Signature**

object.Delete(FullName)

object

DEMFiles
The object or objects this property applies to.

FullName

String; input-only
Full name of the DEM file to be deleted.

Deletes a ContourItem from the Surface.

**Signature**

object.Delete(Id)

object

ContourItems
The object or objects this property applies to.

Id

Long; input-only
The Id of the ContourItem to be deleted.

Deletes the DescriptionKey.

**Signature**
object.Delete(syntax)

object

**DescriptionKeyFile**
The object or objects this property applies to.

syntax

String; input-only
The syntax of the **DescriptionKey** to delete.

---

Deletes the DescriptionKeyFile and removes it from the collection for the project database.

**Signature**

object.Delete(Name)

object

**DescriptionKeyFiles**
The object or objects this property applies to.

Name

String; input-only
The name of the **DescriptionKeyFile** to delete.

---

Deletes a EGProfile and removes it from the EGProfiles collection.

**Signature**

Object.Delete(Type, SurfaceName)

object

**EGProfiles**
The object or objects this property applies to.

Type
eAeccEGProfilesType enum; input-only
The type to delete from the EGProfiles collection.

kEgCenter: Existing ground center profile
kEgLeft: Existing ground left profile
kEgRight: Existing ground right profile

SurfaceName

String; input-only
The name of the surface.

Deletes a Finished Ground Profile.

**Signature**

Object.Delete(Type)

object

**FGProfiles**
The object or objects this property applies to.

Type
eAeccFGProfileType enum; input-only
The type of Finished Ground Profile to delete.

kFgCenter: Finished ground center profile
kFgDitchLeft: Finished ground left ditch profile
kFgDitchRight: Finished ground right ditch profile
kFgLeft1: Finished ground left profile 1
kFgLeft2: Finished ground left profile 2
Finished ground left profile 3
Finished ground left profile 4
Finished ground left profile 5
Finished ground left profile 6
Finished ground left profile 7
Finished ground left profile 8
Finished ground none
Finished ground right profile 1
Finished ground right profile 2
Finished ground right profile 3
Finished ground right profile 4
Finished ground right profile 5
Finished ground right profile 6
Finished ground right profile 7
Finished ground right profile 8

Deletes a Parcel from the Parcels collection.

Signature
object.Delete(Name)

object

**Parcels**
The object or objects this property applies to.

Name

String; input-only
The name of the Parcel to delete.

Deletes a point file from the PointFiles collection.

**Signature**

object.Delete(FullName)

object

**PointFiles**
The object or objects this property applies to.

 FullName

String; input-only
The full name of the **PointFile** to delete.

**Remarks**

Remove will not delete the physical file; it only removes the FileName from the Surface definition.

Deletes a PointGroupName from the PointGroupNames collection.

**Signature**

object.Delete(Name)

object

**PointGroupNames**
The object or objects this property applies to.

Name

String; input-only
The name of the PointGroupName to delete.

---

Deletes a point group from the project database.

**Signature**

object.Delete(Name)

object

**PointGroups**
The object or objects this property applies to.

Name

String; input-only

---

Deletes a specific project

**Signature**

object.Delete(ProjectName)

object

**Projects**
The object or objects this property applies to.

ProjectName

String; input-only
The name of the project to delete.

**Remarks**

The Delete method deletes everything in the <project name> folder, including the drawing files if they are located in the project folder.
No warning dialog is presented. The project is deleted, and not moved to the Windows Recycle Bin.

Deletes the specified prototype.

**Signature**

object.Delete(PrototypeName)

  object

  **Prototypes**
  The object or objects this property applies to.

  ** PrototypeName**
  String; input-only
  The name of the prototype you wish to delete.

**Remarks**

A warning dialog box is displayed, informing you that all files and folders within the prototype folder will be deleted.

Deletes a PVI.

**Signature**

Object.Delete(Station)

  object

  **PVIs**
  The object or objects this property applies to.

  **Station**
  Double; input-only
  The Station of the PVI to delete.

Deletes a station equation from the alignment.
**Signature**

object.Delete(Index)

object

StationEquations
The object or objects this property applies to.

Index

Variant; input-only
The index location in the collection for the member item to query. The index must be either an integer or a string. If an integer, the index must be between 0 and N-1, where N is the number of objects in the collection or selection set.

**Remarks**

If a station equation is deleted, then the indices for the StationEquations collection will change.

---

Deletes the Surface from the Project and from the Surfaces collection.

**Signature**

object.Delete(SurfaceName)

object

Surfaces
The object or objects this property applies to.

SurfaceName

String; input-only
Name of the surface to be deleted.
DifferenceGrid Method

Creates a surface from two existing surfaces using the Grid method.

See Also | Example

Signature

RetVal = object.DifferenceGrid(NewSurface, Surface1, Surface2, X, Y, Rows, Columns, OriginX, OriginY, Angle)

object

- **Surfaces** The object or objects this property applies to.

NewSurface

- String; input-only
  - The name of the new surface to be created.

Surface1

- String; input-only
  - The name of the first surface.

Surface2

- String; input-only
  - The name of the second surface.

X

- Double; input-only
  - The width of the grid cell.

Y
Double; input-only
The height of the grid cell.

Rows
Integer; input-only
The number of rows for the grid.

Columns
Integer; input-only
The number of columns for the grid.

OriginX
Double; input-only
The X coordinate of the grid origin.

OriginY
Double; input-only
The Y coordinate of the grid origin.

Angle
Double; input-only
The rotation angle of the grid.

RetVal

Surface object
The newly added Surface object.
DoubleToStaFormat Method

Given a double, returns a string with the Station Format applied

See Also | Example

**Signature**

RetVal = object.DoubleToStaFormat(Station)

object

*Alignments* The object this property applies to.

Station

Double; read-only
The double to be formatted.

RetVal

String
Station formatted string.

**Remarks**

The Alignments DoubleToStaFormat method differs somewhat from the Alignment [GetStaStrWithEquations](#) method. DoubleToStaFormat does not use Station Equations and is not specific to an Alignment.
EastNorthToXy Method

Given an Easting and Northing, will return the AutoCAD XY

See Also | Example

Signature

RetVal = object.EastNorthToXy(EastNorth)

object

Utility The object or objects this property applies to.

EastNorth

Variant (3 element array of doubles); input-only
The Easting and Northing.

RetVal

Variant (3 element array of doubles)
The coordinates in XY.

Remarks

Xy must be declared as a variant and not as an array. To access the X and Y, subscript the variant ( var(0), var(1)). The third element (elevation) is unchanged.
ElevationAt Method (Civil Engineering Feature)

Given a station, returns the elevation..

**Signature**

RetVal = object.ElevationAt(Station)

- **object**
  - `EGProfile, FGProfile` The object or objects this property applies to.

- **Station**
  - Double, input-only
  - The station to query for.

- **RetVal**
  - Double
  - The elevation for the station.
**ExternalStaToInternal Method**

Given an external Station value (one using Station Equations), returns the internal Station value

**Signature**

RetVal = object.ExternalStaToInternal(Station)

object

*Alignment* The object this property applies to.

Station

Double; read-only
The external station.
Variant (array of doubles)

RetVal

Variant
An array of internal station values.

**Remarks**

An External Station value represents one or more internal Station values. These values are returned in an array of doubles.
FindAllConnectingEdges Method

Returns all edges for a given Surface point.

See Also | Example

Signature

RetVal = object.FindAllConnectingEdges(Easting, Northing)

object

Surface The object or objects this property applies to.

Easting

Double; input-only
The exact Easting of the Surface point.

Northing

Double; input-only
The exact Northing of the Surface point.

RetVal

Variant (array of doubles)
An array of 3D points.

Remarks

FindAllConnectingEdges expects the Easting / Northing to be a Surface point. If not, the function will return with an error. The exact coordinates can be found using the Surface FindPoint method or by using a TinPoint
The first element in the array will be the coordinate passed in. The following items in the array are all of the point that radiate from the selected point. All edges are returned in counter-clockwise order.
FindAllFaces Method

Redefines the collection with all faces on the surface.

See Also | Example

Signature

object.FindAllFaces()

object

Faces The object or objects this property applies to.

Remarks

Use this method to reset the collection to all faces in the surface. The SearchType property will be set to kNoSearch since no filter is being applied.
FindConnectingEdge Method

Given an Surface point, retrieves the closest edge.

See Also |Example

Signature

RetVal = object.FindConnectingEdge(Easting, Northing)

object

Surface The object or objects this property applies to.

Easting

Double; input-only
Easting of the Surface point.

Northing

Double; input-only
Northing of the Surface point.

RetVal

Variant (6 element array of doubles)
An array of two 3D points.

Remarks

FindConnectingEdge will return the closest edge to the input point.
FindFace Method

Given an Easting / Northing coordinate, redefines the collection with a single Face.

See Also | Example

Signature

object.FindFace(Easting, Northing)

object

Faces The object or objects this property applies to.

Easting

Double; input-only
The Easting coordinate to query.

Northing

Double; input-only
The Northing coordinate to query.
**FindPath Method**

Give two coordinates, reads in all of the Faces that lie on that line.

See Also | Example

**Signature**

object.FindPath(StartEasting, StartNorthing, EndEasting, EndNorthing)

object

- **Faces** The object or objects this property applies to.

StartEasting

- Double; input-only
  - The first point's Easting coordinate.

StartNorthing

- Double; input-only
  - The first point's Northing coordinate.

EndEasting

- Double; input-only
  - The second point's Easting coordinate.

EndNorthing

- Double; input-only
  - The second point's Northing coordinate.

**Remarks**

FindPath will act as a filter for the Faces collection. Given two points, it
will return all of the faces that the line crosses. The Faces collection now contains only those faces. The SearchType will be set to kSearchByPath.
FindPoint Method

Given an Easting / Northing, returns the closest Surface point.

See Also | Example

Signature

RetVal = object.FindPoint(Easting, Northing)

object

Surface The object or objects this property applies to.

Easting

Double; input-only
The Easting of the Surface point.

Northing

Double; input-only
The Northing of the Surface point.

RetVal

Variant (3 element array of doubles)
A 3D point representing the Surface point.
GetBoundingBox Method

Gets the bounding box for the surface.

See Also | Example

Signature

object.GetBoundingBox(LowerLeftPoint, UpperRightPoint)

object

Surface The object or objects this property applies to.

LowerLeftPoint

Variant (3 element array of doubles); output-only
The lower left coordinate for the bounding box in Easting, Northing, Elevation format.

UpperRightPoint

Variant (3 element array of doubles); output-only
The upper right coordinate for the bounding box in Easting, Northing, Elevation format.

Remarks

All coordinates for the surface will lie within the bounding box. The surface must be open and built before using the GetBoundingBox method.
GetDouble Method

Gets the specified preference setting.

Signature: Overview

- PreferencesAlignment
- PreferencesCogo
- PreferencesCrossSection (Civil Engineering Feature)
- PreferencesProfile (Civil Engineering Feature)
- PreferencesSurface

Gets the specified preference setting for Alignments.

Signature

RetVal = object.GetDouble(Setting)

object

PreferencesAlignment The object or objects this property applies to.

Setting
eAeccPrefAlignDouble enum; input-only.
Specifies the setting to return.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kOffsetLeftDistOuter</td>
<td>Offset alignment - left distance outer</td>
</tr>
<tr>
<td>kOffsetRightDistOuter</td>
<td>Offset alignment - right distance outer</td>
</tr>
<tr>
<td>kOffsetLeftDistSecond</td>
<td>Offset alignment - left distance second</td>
</tr>
<tr>
<td>kOffsetRightDistSecond</td>
<td>Offset alignment - right distance second</td>
</tr>
<tr>
<td>kOffsetLeftDistThird</td>
<td>Offset alignment - left distance third</td>
</tr>
<tr>
<td>kOffsetRightDistThird</td>
<td>Offset alignment - right distance third</td>
</tr>
<tr>
<td>kOffsetLeftDistInner</td>
<td>Offset alignment - left distance inner</td>
</tr>
<tr>
<td>kOffsetRightDistInner</td>
<td>Offset alignment - right distance inner</td>
</tr>
<tr>
<td>kStationTickIncrement</td>
<td>Station labels tick increment</td>
</tr>
<tr>
<td>kStationLabelIncrement</td>
<td>Station labels increment</td>
</tr>
<tr>
<td>kStationLabelOffset</td>
<td>Station labels offset</td>
</tr>
</tbody>
</table>

RetVal

Double
The value of the setting.
Gets the specified preference setting for Cogo.

**Signature**

RetVal = object.GetDouble(Setting)

object

*PreferencesCogo*
The object or objects this property applies to.

Setting

eAeccPrefCogoDouble enum; input-only. Specifies the setting to return.

kPntCreateDefaultElev: Create default elevation
kPntInsertFixedElev: Insert fixed elevation
kPntInsertNoActualElev: Insert no elevation
kPntMarkerSize: Point marker size
kPntTextSize: Point text size
kPntTextRotation: Point text rotation

RetVal

Double
The value of the setting.

---

Gets the specified preference setting for Cogo.

**Signature**

RetVal = object.GetDouble(Setting)
object

**PreferencesCrossSection**
The object or objects this property applies to.

**Setting**

eAeccPrefCrossSectionDouble enum; input-only. Specifies the setting to return.

kElevationInc: Elevation increment

kOffsetInc: Offset increment

**RetVal**

Double
The value of the setting.

Gets the specified preference setting for Vertical Alignments.

**Signature**

RetVal = object.GetDouble(Setting)

object

**PreferencesProfile**
The object or objects this property applies to.

**Setting**

eAeccPrefProfileDouble enum; input-only. Specifies the setting to return.

kLabelInc: Label increment

kTangentLabelInc: Tangent label increment

kVerticalCurveLabelInc: Vertical curve label increment
**kPassingSightDistEye:** Passing sight eye height distance

**kPassingSightDistObj:** Passing sight object height distance

**kStoppingSightDistEye:** Stopping sight eye height distance

**kStoppingSightDist_obj:** Stopping sight object height distance

**kKCrest:** K minimum for crest

**kKSag:** K minimum for sag

**kOverlap:** Overlap

**RetVal**

`Double`

The value of the setting.

---

Gets the specified preference setting for Surfaces.

**Signature**

RetVal = object.GetDouble(Setting)

**object**

`PreferencesSurface`

The object or objects this property applies to.

**Setting**

`eAeccPrefSurfaceDouble` enum; input-only.

Specifies the setting to return.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kContWeedDist:</td>
<td>Contour weeding distance</td>
</tr>
<tr>
<td>kContWeedAngle:</td>
<td>Contour weeding angle</td>
</tr>
<tr>
<td>kContWeedSuppDist:</td>
<td>Contour supplementing distance</td>
</tr>
<tr>
<td>kContWeedSuppBulge:</td>
<td>Contour supplementing bulge</td>
</tr>
<tr>
<td>kVertExaggerationFactor:</td>
<td>Surface vertical exaggeration factor</td>
</tr>
<tr>
<td>kElevRangeBegVal1:</td>
<td>Elevation begin range - range 1</td>
</tr>
<tr>
<td>kElevRangeBegVal2:</td>
<td>Elevation begin range - range 2</td>
</tr>
<tr>
<td>kElevRangeBegVal3:</td>
<td>Elevation begin range - range 3</td>
</tr>
<tr>
<td>kElevRangeBegVal4:</td>
<td>Elevation begin range - range 4</td>
</tr>
<tr>
<td>kElevRangeBegVal5:</td>
<td>Elevation begin range - range 5</td>
</tr>
<tr>
<td>kElevRangeBegVal6:</td>
<td>Elevation begin range - range 6</td>
</tr>
<tr>
<td>kElevRangeBegVal7:</td>
<td>Elevation begin range - range 7</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td><code>kElevRangeBegVal8:</code></td>
<td>Elevation begin range - range 8</td>
</tr>
<tr>
<td><code>kElevRangeBegVal9:</code></td>
<td>Elevation begin range - range 9</td>
</tr>
<tr>
<td><code>kElevRangeBegVal10:</code></td>
<td>Elevation begin range - range 10</td>
</tr>
<tr>
<td><code>kElevRangeBegVal11:</code></td>
<td>Elevation begin range - range 11</td>
</tr>
<tr>
<td><code>kElevRangeBegVal12:</code></td>
<td>Elevation begin range - range 12</td>
</tr>
<tr>
<td><code>kElevRangeBegVal13:</code></td>
<td>Elevation begin range - range 13</td>
</tr>
<tr>
<td><code>kElevRangeBegVal14:</code></td>
<td>Elevation begin range - range 14</td>
</tr>
<tr>
<td><code>kElevRangeBegVal15:</code></td>
<td>Elevation begin range - range 15</td>
</tr>
<tr>
<td><code>kElevRangeBegVal16:</code></td>
<td>Elevation begin range - range 16</td>
</tr>
<tr>
<td><code>kElevRangeEndVal1:</code></td>
<td>Elevation end range - range 1</td>
</tr>
<tr>
<td><code>kElevRangeEndVal2:</code></td>
<td>Elevation end range - range 2</td>
</tr>
<tr>
<td><code>kElevRangeEndVal3:</code></td>
<td>Elevation end range - range 3</td>
</tr>
</tbody>
</table>
kElevRangeEndVal4: Elevation end range - range 4
kElevRangeEndVal5: Elevation end range - range 5
kElevRangeEndVal6: Elevation end range - range 6
kElevRangeEndVal7: Elevation end range - range 7
kElevRangeEndVal8: Elevation end range - range 8
kElevRangeEndVal9: Elevation end range - range 9
kElevRangeEndVal10: Elevation end range - range 10
kElevRangeEndVal11: Elevation end range - range 11
kElevRangeEndVal12: Elevation end range - range 12
kElevRangeEndVal13: Elevation end range - range 13
kElevRangeEndVal14: Elevation end range - range 14
kElevRangeEndVal15: Elevation end range - range 15
kElevRangeEndVal16:  Elevation end range - range 16

kSurfDisplayBaseElev:  Surface - display base elevation

kContCreateMinElev:  Contours create - minimum contour starting elevation

kContCreateMaxElev:  Contours create - maximum contour starting elevation

kContCreateMinorInterval:  Contours create - minor contour interval

kContCreateMinorWidth:  Contours create - minor contour width

kContCreateMajorInterval:  Contours create - major contour interval

kContCreateMajorWidth:  Contours create - major contour width

kContLabelElevIncrement:  Contours - label elevation increment

kContLabelSpacingDist:  Contours - Label spacing distance

kWShedMinDepth:  Water Sheds - minimum depth
kWShedMinArea: minimum area

RetVal

Double
The value of the setting.
GetElevation Method

Given a Northing and Easting, return the elevation.

See Also | Example

Signature

RetVal = object.GetElevation(Easting, Northing)

object

Surface The object or objects this property applies to.

Easting

Double; input-only
The Easting coordinate to query.

Northing

Double; input-only
The Northing coordinate to query.

RetVal

Double
The elevation for the Easting / Northing.

Remarks

If the coordinates are within the bounds of the Surface, the elevation will be returned. If the coordinates lie outside of the Surface or lie in a hole in the Surface, then -1e20 will be returned.
GetInteger Method

Gets the specified preference setting.

Signature: Overview

| PreferencesAlignment |
| PreferencesCogo      |
| PreferencesCrossSection (Civil Engineering Feature) |
| PreferencesParcel    |
| PreferencesProfile   (Civil Engineering Feature) |
| PreferencesSurface   |

Gets the specified preference setting for Alignments.

Signature

RetVal = object.GetInteger(Setting)

object

PreferencesAlignment The object or objects this property applies to.

Setting
`eAeccPrefAlignInt enum; input-only.
Specifies the setting to return.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>kStationFmtUseStationFormat</code></td>
<td>Station format - use</td>
</tr>
<tr>
<td><code>kStationFmtUseNegParen</code></td>
<td>Station format - use negative parentheses</td>
</tr>
<tr>
<td><code>kStationFmtDropDecimal</code></td>
<td>Station format - drop decimal</td>
</tr>
<tr>
<td><code>kStationFmtLeadZeros</code></td>
<td>Station format - use lead zeros</td>
</tr>
<tr>
<td><code>kStationFmtCharPos</code></td>
<td>Station format - character position</td>
</tr>
<tr>
<td><code>kStationFmtFieldWidth</code></td>
<td>Station format - field width</td>
</tr>
<tr>
<td><code>kStationFmtDecimalPrec</code></td>
<td>Station format - decimal precision</td>
</tr>
<tr>
<td><code>kSpiralType</code></td>
<td>Spiral type</td>
</tr>
<tr>
<td><code>kOffsetToggle</code></td>
<td>Offset toggle</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RetVal</th>
<th>Integer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The value of the setting.</td>
</tr>
</tbody>
</table>

Gets the specified preference setting for Cogo.

**Signature**
RetVal = object.GetInteger(Setting)

object

**PreferencesCogo**
The object or objects this property applies to.

Setting

eAeccPrefCogoInt enum; input-only.
Specifies the setting to return.

- **kPntCreateNextNum:** Next point number
- **kPntInsertAsCreated:** Insert points into drawing as created
- **kPntCreateNumSequential:** Point number sequential
- **kPntCreateElevMode:** Point - create elevation mode
- **kPntCreateDescMode:** Point - create description mode
- **kPntInsertAtElevation:** Point - insert at elevation
- **kPntInsertAutoLabel:** Point - insert autolabel
- **kPntUpdateAllowMove:** Point update - allow point move
- **kPntUpdateDbAfterMove:** Point update - update database after move
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kPntUpdateReuniteSym</td>
<td>Point update - reunite symbol</td>
</tr>
<tr>
<td>kPntUpdateCheckDbOnOpen</td>
<td>Point update - check database on open</td>
</tr>
<tr>
<td>kPntCoordDispType</td>
<td>Point - coordinate display type</td>
</tr>
<tr>
<td>kPntCoordEcho</td>
<td>Point - echo coordinates</td>
</tr>
<tr>
<td>kDscKeySearchOrder</td>
<td>Description Key - search order</td>
</tr>
<tr>
<td>kDscKeyExtendedSearch</td>
<td>Description Key - extended search order</td>
</tr>
<tr>
<td>kDscKeyParamMatch</td>
<td>Description Key - Parameter match</td>
</tr>
<tr>
<td>kPntPrefCmdLineListOn</td>
<td>Point - Command line list on</td>
</tr>
<tr>
<td>kPntPrefCmdLineGroupOn</td>
<td>Point - command line group on</td>
</tr>
<tr>
<td>kPntPrefAutoRegen</td>
<td>Point - auto regen</td>
</tr>
<tr>
<td>kPntPrefSortListRem</td>
<td>Point - sort list</td>
</tr>
<tr>
<td>kPntPrefSortListDuplicate</td>
<td>Point - sort list duplicate</td>
</tr>
</tbody>
</table>
kPntMarkerAcadPoint: Point marker - AutoCAD point
kPntMarkerBox: Point marker - box
kPntMarkerCircle: Point marker - circle
kPntMarkerStyle: Point marker - style
kPntMarkerAlignWithText: Point marker - align with text
kPntNumVisible: Point - number is visible
kPntElevVisible: Point - elevation is visible
kPntDescVisible: Point - description is visible
kPntShowFullDesc: Point - show full description
kPntNumColor: Point - number color
kPntElevColor: Point - elevation color
kPntDescColor: Point - description color
kPntAutoLeader: Point - auto leader

RetVal
Integet
The value of the setting.

Gets the specified preference setting for Cross Sections.

**Signature**

RetVal = object.GetInteger(Setting)

object

*PreferencesCrossSection*
The object or objects this property applies to.

Setting

eAeccPrefCrossSectionInt enum; input-only.
Specifies the setting to return.

  *kDatumLayerOn*: Datum layer visibility
  *kEGLabelPrecision*: Existing ground label precision
  *kEGLayerOn*: Existing ground layer visibility
  *kElevationLabelInc*: Elevation label increment
  *kElevationPrecision*: Elevation precision
  *kFGLabelPrecision*: Finished ground label precision
  *kOffsetLabelInc*: Offset label increment
  *kOffsetPrecision*: Offset precision
kROWLinesLayerOn: Right of Way line layer visibility
kRowsAboveMax: Rows above maximum
kRowsBelowDatum: Rows below datum
kSectionGridLayerOn: Section grid layer visibility
kSectionGridTextLayerOn: Section grid text layer visibility
kTemplateLayerOn: Template layer visibility

RetVal

Integer
The value of the setting.

Gets the specified preference setting for Parcels.

**Signature**

RetVal = object.GetInteger(Setting)

object

[PreferencesParcel](#)
The object or objects this property applies to.

Setting

eAeccPrefParcelsInt enum; input-only.
Specifies the setting to return.

kDefineAsSized: Define parcel as sized
kCheckAcrossChord: Map check across chord
kTruncateAreaLabels: Truncate area labels
kAutoLabelPlacement: Automatic label placement
kIncludeParcelLines: Include parcel lines on import
kNumberLabelsOn: Parcel number label visibility
kSequentialOn: Sequential parcel numbering
kNextParcelNumber: Next parcel number
kSqUnitLabelsOn: Square unit label visibility
kSqUnitPrecision: Square unit precision
kAreaUnitLabelsOn: Area unit label visibility
kAreaUnitPrecision: Area unit precision

RetVal

    Integer
    The value of the setting.

---

Gets the specified preference setting for Vertical Alignments.

**Signature**

RetVal = object.GetInteger(Setting)
object

PreferencesProfile
The object or objects this property applies to.

Setting

eAeccPrefProfileInt enum; input-only.
Specifies the setting to return.

kEgPrecision: Existing ground label precision

kFgPrecision: Finished ground label precision

kGridDsp: 0: Grid creation is off
1: Grid creation is on

klr_on: 0: Left to right creation is set
1: Right to left creation is set

RetVal

Integer
The value of the setting.

---

Gets the specified preference setting for Surfaces.

Signature

RetVal = object.GetInteger(Setting)

object

PreferencesSurface
The object or objects this property applies to.

Setting

eAeccPrefSurfaceInt enum; input-only.
Specifies the setting to return.
kviewtype: View type
kRangeNumber: Range number
kfltdel: Fault delete
krngs_type: Ranges type
kRange1Color: Range color 1
kRange2Color: Range color 2
kRange3Color: Range color 3
kRange4Color: Range color 4
kRange5Color: Range color 5
kRange6Color: Range color 6
kRange7Color: Range color 7
kRange8Color: Range color 8
kRange9Color: Range color 9
kRange10Color: Range color 10
kRange11Color: Range color 11
kRange12Color: Range color 12
kRange13Color: Range color 13
kRange14Color: Range color 14
<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kRange15Color:</td>
<td>Range color 15</td>
</tr>
<tr>
<td>kRange16Color:</td>
<td>Range color 16</td>
</tr>
<tr>
<td>kGraphicsColor:</td>
<td>Graphics color</td>
</tr>
<tr>
<td>kContSmoothFactor:</td>
<td>Contour - smoothing factor</td>
</tr>
<tr>
<td>kContSmoothType:</td>
<td>Contour - smoothing type</td>
</tr>
<tr>
<td>kContLabelPrecision:</td>
<td>Contour - label precision</td>
</tr>
<tr>
<td>kContCreateAsPoly:</td>
<td>Contour - create as polyline</td>
</tr>
<tr>
<td>kContLabelMultipleToggle:</td>
<td>Contour - label multiple</td>
</tr>
<tr>
<td>kContStyleMgrLastUsedPage:</td>
<td>Contour - style manager last used page</td>
</tr>
<tr>
<td>kWShedExceedBothCriteria:</td>
<td>Watershed - exceed both criteria</td>
</tr>
<tr>
<td>kWShedErasePrevious:</td>
<td>Watershed - erase previous</td>
</tr>
<tr>
<td>kWShedDisplayNumber:</td>
<td>Watershed - display number</td>
</tr>
<tr>
<td>kWShedDisplayNumber:</td>
<td>Watershed - use</td>
</tr>
</tbody>
</table>
RetVal

Integer
The value of the setting.
GetLayerName Method (Civil Engineering Feature)

Returns the layer names of the alignment cross section or alignment profile.

See Also | Example

**Signature: Overview**

- CrossSectionBlock (Civil Engineering Feature)
- ProfileBlock (Civil Engineering Feature)

Returns the layer names of the alignment cross section.

**Signature**

RetVal = object.GetLayerName(val)

**object**

CrossSectionBlock The object or objects this property applies to.

**val**

eAeccCrossSectionBlockLayer enum; input-only. Specifies the layer name to return.

- kCrossSectionEGTextLayer: Existing grade text layer
- kCrossSectionEGLayer: Existing grade layer
RetVal

String
Returns the Layer Name of the CrossSectionBlock.

Returns the layer names of the alignment profile.

**Signature**

RetVal = object.GetLayerName(val)

object

ProfileBlock
The object or objects this property applies to.

val
eAeccProfileBlockLayer enum; input-only.
Specifies the layer name to return.

kBaseVertical: Existing ground base vertical layer

kBaseHorizontal: Existing ground base horizontal layer

kDitchLeft: Finished ground ditch left layer
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kDitchRight</td>
<td>Finished ground ditch right layer</td>
</tr>
<tr>
<td>kEGC</td>
<td>Existing ground center surface layer</td>
</tr>
<tr>
<td>kEGCText</td>
<td>Existing ground center surface text layer</td>
</tr>
<tr>
<td>kEGL</td>
<td>Existing ground left surface layer</td>
</tr>
<tr>
<td>kEGLText</td>
<td>Existing ground left surface text layer</td>
</tr>
<tr>
<td>kEGR</td>
<td>Existing ground right surface layer</td>
</tr>
<tr>
<td>kEGRText</td>
<td>Existing ground right surface text layer</td>
</tr>
<tr>
<td>kFGC</td>
<td>Finished ground center</td>
</tr>
<tr>
<td>kFGCText</td>
<td>Finished ground center text</td>
</tr>
<tr>
<td>kFGL1</td>
<td>Finished ground left trans 1 layer</td>
</tr>
<tr>
<td>kFGL2</td>
<td>Finished ground left trans 2 layer</td>
</tr>
<tr>
<td>kFGL3</td>
<td>Finished ground left trans 3 layer</td>
</tr>
<tr>
<td>kFGL4</td>
<td>Finished ground left trans 4 layer</td>
</tr>
</tbody>
</table>
kFGL5: Finished ground left trans 5 layer
kFGL6: Finished ground left trans 6 layer
kFGL7: Finished ground left trans 7 layer
kFGL8: Finished ground left trans 8 layer
kFGR1: Finished ground right trans 1 layer
kFGR2: Finished ground right trans 2 layer
kFGR3: Finished ground right trans 3 layer
kFGR4: Finished ground right trans 4 layer
kFGR5: Finished ground right trans 5 layer
kFGR6: Finished ground right trans 6 layer
kFGR7: Finished ground right trans 7 layer
kFGR8: Finished ground right trans 8 layer
kGrid: Existing ground grid layer
kGridText: Existing ground text layer
kStationText: Existing ground station text

RetVal
String
Returns the Layer Name of the ProfileBlock.
GetStaStrWithEquations Method

Given an internal station value, returns that station as a formatted string with Station Equations applied.

See Also | Example

Signature

RetVal = object.GetStaStrWithEquations(Station)

object

Alignment The object or objects this property applies to.

Station

Double; input-only
Internal station value.

RetVal

String
The formatted station string.
GetStaWithEquations Method

Given an internal station value, returns that station value with Station Equations applied.

See Also | Example

Signature

RetVal = object.GetStaWithEquations(Station)

object

Alignment The object or objects this property applies to.

Station

Double; input-only
Internal station value.

RetVal

Double
The station number.
GetString Method

Gets the specified preference setting.

See Also | Example

Signature: Overview

- PreferencesAlignment
- PreferencesCogo
- PreferencesCrossSection (Civil Engineering Feature)
- PreferencesParcel
- PreferencesProfile (Civil Engineering Feature)
- PreferencesSurface

Gets the specified preference setting for Alignments.

Signature

RetVal = object.GetString(Setting)

object

PreferencesAlignment The object or objects this property applies to.

Setting
eAeccPrefAlignString enum; input-only. Specifies the setting to return.

- **kStationFmtDecChar:** Station format - decimal character
- **kStationFmtStaChar:** Station format - station character
- **kAlignLayerPrefix:** Alignment layer prefix
- **kLabelStaEqu Ahead:** Label station equation ahead
- **kLabelStaEqu Back:** Label station equation back
- **kLabelPT:** Label PT
- **kLabelPC:** Label PC
- **kLabelCPI:** Label CPI
- **kLabelPCC:** Label PCC
- **kLabelPRC:** Label PRC
- **kLabelTS:** Label TS
- **kLabelSC:** Label SC
- **kLabelCS:** Label CS
- **kLabelST:** Label ST
- **kLabelSPI:** Label SPI
kLabelCC: Label CC
kLabelPI: Label PI
kLabelSS: Label SS
kOffsetLeftLayerOuter: Offsets layer - left outer
kOffsetLeftLayerSecond: Offsets layer - left second
kOffsetLeftLayerThird: Offsets layer - left third
kOffsetLeftLayerInner: Offsets layer - left inner
kOffsetLeftNameOuter: Offsets name - left outer
kOffsetRightNameOuter: Offsets name - right name
kOffsetLeftNameSecond: Offsets name - left second
kOffsetRightNameSecond: Offsets name - right second
kOffsetLeftNameThird: Offsets name - left third
kOffsetRightNameThird: Offsets name - right third
kOffsetLeftNameInner: Offsets name - left inner
kOffsetRightNameInner: Offsets name - right inner

kOffsetNamePrefix: Offsets name - prefix

kStationLayer: Station layer

kStationPntLayer: Station point layer

kStationEquLayer: Station equations layer

RetVal

String
The value of the setting.

-----------------------------------------------------------

Gets the specified preference setting for Cogo.

**Signature**

RetVal = object.GetString(Setting)

object

PreferencesCogo
The object or objects this property applies to.

Setting
eAeccPrefCogoString enum; input-only.
Specifies the setting to return.

kPntCreateDefaultDesc: Point - Create default description

kDscKeySymPath: Description Key symbol path

kPntTextStyle: Point - text style
RetVal

String
The value of the setting.

Gets the specified preference setting for Cross Sections.

**Signature**

RetVal = object.GetString(Setting)

object

PreferencesCrossSection
The object or objects this property applies to.

Setting
eAeccPrefCrossSectionString enum; input-only.
Specifies the setting to return.

kDatumLayer: Datum layer
kEGLayer: Existing grade layer
kROWLinesLayer: Right of Way lines layer
kSectionGridLayer: Section grid layer
kSectionGridTextLayer: Section grid text layer
kTemplateLayer: Template layer
kTemplatePath: Cross section template path

RetVal
String
The value of the setting.

Gets the specified preference setting for Parcels.

**Signature**

RetVal = object.GetString(Setting)

object

PreferencesParcel
The object or objects this property applies to.

Setting

eAeccPrefParcelsString enum; input-only.
Specifies the setting to return.

kNumberPrefix: Parcel numbering prefix
kNumberTextStyle: Parcel numbering text style
kSqUnitTextStyle: Square unit text style
kSqUnitSuffix: Square unit suffix
kAreaUnitTextStyle: Area unit text style
kAreaUnitSuffix: Area unit suffix
kParcelLayer: Parcel layer
kLabelLayer: Parcel label layer

RetVal
String
The value of the setting.

Gets the specified preference setting for Vertical Alignments.

**Signature**

RetVal = object.GetString(Setting)

object

PreferencesProfile
The object or objects this property applies to.

Setting

eAeccPrefProfilelString enum; input-only.
Specifies the setting to return.

kTextLayer: Existing ground station text layer

kVertGridLayer: Existing ground vertical grid layer

kBaseGridLayer: Existing ground base grid layer

kEgLayer: Existing ground center layer

kEgLeftLayer: Existing ground left layer

kEgRightLayer: Existing ground right layer

kFgLayer: Finished ground center layer

kGridLayer: Existing ground grid layer

kGridTextLayer: Existing ground grid text layer
kEgtLayer: Existing ground center text layer
kEgLeftLayer: Existing ground left text layer
kEgRightLayer: Existing ground right text layer
kDitchLeftLayer: Finished ground ditch left layer
kDitchRightLayer: Finished ground ditch right layer
kFgtLayer: Finished ground text layer
kBvcs: Beginning vertical curve station label text
kBvce: Beginning vertical curve elevation text
kEvcs: End vertical curve station label text
kEvce: End vertical curve elevation text
kProfileDir: Sample profile direction
kFgLeftLayer1: Finished ground left 1 trans layer
kFgLeftLayer2: Finished ground left 2 trans layer
kFgLeftLayer3: Finished ground left 3 trans layer
<table>
<thead>
<tr>
<th>Layer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kFgLeftLayer4:</td>
<td>Finished ground left 4 trans layer</td>
</tr>
<tr>
<td>kFgLeftLayer5:</td>
<td>Finished ground left 5 trans layer</td>
</tr>
<tr>
<td>kFgLeftLayer6:</td>
<td>Finished ground left 6 trans layer</td>
</tr>
<tr>
<td>kFgLeftLayer7:</td>
<td>Finished ground left 7 trans layer</td>
</tr>
<tr>
<td>kFgLeftLayer8:</td>
<td>Finished ground left 8 trans layer</td>
</tr>
<tr>
<td>kFgRightLayer1:</td>
<td>Finished ground right 1 trans layer</td>
</tr>
<tr>
<td>kFgRightLayer2:</td>
<td>Finished ground right 2 trans layer</td>
</tr>
<tr>
<td>kFgRightLayer3:</td>
<td>Finished ground right 3 trans layer</td>
</tr>
<tr>
<td>kFgRightLayer4:</td>
<td>Finished ground right 4 trans layer</td>
</tr>
<tr>
<td>kFgRightLayer5:</td>
<td>Finished ground right 5 trans layer</td>
</tr>
<tr>
<td>kFgRightLayer6:</td>
<td>Finished ground right 6 trans layer</td>
</tr>
<tr>
<td>kFgRightLayer7:</td>
<td>Finished ground right 7 trans layer</td>
</tr>
</tbody>
</table>
RetVal

String
The value of the setting.

Gets the specified preference setting for Surfaces.

**Signature**

RetVal = object.GetString(Setting)

**object**

`PreferencesSurface`
The object or objects this property applies to.

**Setting**

eAeccPrefSurfaceString enum; input-only.
Specifies the setting to return.
kSlopeRangeBegVal1: Slope range - begin 1
kSlopeRangeBegVal2: Slope range - begin 2
kSlopeRangeBegVal3: Slope range - begin 3
kSlopeRangeBegVal4: Slope range - begin 4
kSlopeRangeBegVal5: Slope range - begin 5
kSlopeRangeBegVal6: Slope range - begin 6
kSlopeRangeBegVal7: Slope range - begin 7
kSlopeRangeBegVal8: Slope range - begin 8
kSlopeRangeBegVal9: Slope range - begin 9
kSlopeRangeBegVal10: Slope range - begin 10
kSlopeRangeBegVal11: Slope range - begin 11
kSlopeRangeBegVal12: Slope range - begin 12
kSlopeRangeBegVal13: Slope range - begin 13
kSlopeRangeBegVal14: Slope range - begin 14
kSlopeRangeBegVal15: Slope range - begin 15
kSlopeRangeBegVal16: Slope range - begin 16
kSurfaceLayer: Surface layer
kBoundaryLayer: Boundary layer
kRange1Layer: Range layer 1
kRange2Layer: Range layer 2
kRange3Layer: Range layer 3
kRange4Layer: Range layer 4
kRange5Layer: Range layer 5
kRange6Layer: Range layer 6
kRange7Layer: Range layer 7
kRange8Layer: Range layer 8
kRange9Layer: Range layer 9
kRange10Layer: Range layer 10
<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kRange11Layer:</td>
<td>Range layer 11</td>
</tr>
<tr>
<td>kRange12Layer:</td>
<td>Range layer 12</td>
</tr>
<tr>
<td>kRange13Layer:</td>
<td>Range layer 13</td>
</tr>
<tr>
<td>kRange14Layer:</td>
<td>Range layer 14</td>
</tr>
<tr>
<td>kRange15Layer:</td>
<td>Range layer 15</td>
</tr>
<tr>
<td>kRange16Layer:</td>
<td>Range layer 16</td>
</tr>
<tr>
<td>kBreakLineLayer:</td>
<td>BreakLine layer</td>
</tr>
<tr>
<td>kLayerPrefix:</td>
<td>Surface Layer prefix</td>
</tr>
<tr>
<td>kContCreateMinorLayer:</td>
<td>Contour - create minor layer</td>
</tr>
<tr>
<td>kContCreateMajorLayer:</td>
<td>Contour - create major layer</td>
</tr>
<tr>
<td>kContourStyle:</td>
<td>Contour - style</td>
</tr>
<tr>
<td>kWShedLayerBoundPnt:</td>
<td>Watershed layer - bounding points</td>
</tr>
<tr>
<td>kWShedLayerBoundSeg:</td>
<td>Watershed layer - bounding segments</td>
</tr>
<tr>
<td>kWShedLayerNonBound:</td>
<td>Watershed layer - nonbound</td>
</tr>
</tbody>
</table>
kWShedLayerDepression: Watershed layer - depression

kWShedLayerFlatArea: Watershed layer - flat area

kWShedLayerMultiDrain: Watershed layer - multidrain

kWShedLayerLocalMin: Watershed layer - local minimum

kWShedLayerDrain: Watershed layer - drain

kWShedLayerMultiDrainNotch: Watershed layer - multidrain notch

RetVal

String
The value of the setting.
Import Method

Draws the alignment, cross section, existing ground profile, parcel, or surface into the current drawing.

Signature: Overview

- Alignment
- CrossSection (Civil Engineering Feature)
- EGProfile (Civil Engineering Feature)
- Parcel
- Surface

Draws an alignment into the current drawing.

Signature

object.Import()

object

Alignment The object or objects this property applies to.

Remarks

Import will draw the alignment in the current drawing using the current
layer. When an alignment is imported, external data is stored on the AutoCAD entities and the entity ids are stored externally. If you want to be able to select an alignment from a drawing, you need to use the Import command. If you draw the alignment using your own routines, this extra information will not be supplied.

Draws an alignment cross section into the current drawing.

**Signature**

object.Import(Coordinates[, bBlockOnly])

object

**CrossSection**
The object or objects this property applies to.

Coordinates

Variant (array of doubles); input-only
The starting point in X, Y format.

bBlockOnly

Boolean; input only; optional
True: Only the block is inserted.
False: The cross section is drawn along with the block (default).

Draws the existing ground profile as a Quick Profile in the current drawing.

**Signature**

object.Import(Coordinates[, StartingStation][, EndingStation][, DatumElevation][, VerticalScale][, bLeftToRight][, bBlockOnly])

object

**EGProfile**
The object or objects this property applies to.
Coordinates

Variant (2 element array of doubles); input-only
The starting point in X, Y format.

StartingStation

Double; input only; optional
The starting station of the alignment. The default is the StartingStation of the Alignment object.

EndingStation

Double; input only; optional
The ending station of the alignment. The default is the EndingStation of the Alignment object.

DatumElevation

Double; input only; optional
The elevation of the base line for the profile. The default value is calculated using the minimum elevation of the profile.

VerticalScale

Double; input only; optional
The vertical scale applied to the profile. The default value is the VerticalScale of the DatabasePreferences object.

bLeftToRight

Boolean; input only; optional
True: The profile is drawn left to right (default).
False: The profile is drawn right to left.

bBlockOnly

Boolean; input only; optional
True: Only the block is inserted.
False: The profile is drawn along with the block (default).

Remarks
You may import a profile for center, left or right existing ground. If you import more than 1 profile for a given alignment, you can identify it using the Coordinates property of the ProfileBlock object (note that the Land Desktop commands do not support more than one profile per alignment).

Draws a parcel into the current drawing.

**Signature**

object.Import()

object

Parcel

The object or objects this property applies to.

Draws a surface in the current drawing as 3D surfaces.

**Signature**

object.Import()

object

Surface

The object or objects this property applies to.

**Remarks**

The Import function is not supported for Volume Surfaces (Composite/Diff Grid).
InstantGrade Method

Given a station, returns the instantaneous grade and the algebraic difference.

See Also | Example

Signature: Overview

- EGProfile (Civil Engineering Feature)
- FGProfile (Civil Engineering Feature)

Signature

object.InstantGrade(Station, Grade)

object

EGProfile The object or objects this property applies to.

Station

Double; input-only
The station to query for.

Grade

Double; output-only
The grade at the station.
object.InstantGrade(Station, Grade, Diff)

object

FGProfile
The object or objects this property applies to.

Station

Double; input-only
The station to query for.

Grade

Double; output-only
The grade at the station.

Diff

Double; output-only
The algebraic difference between the two grades.

Remarks

The algebraic difference will only exist for a vertical curve.
**Item Method**

Gets the member object at a given index in a collection.

[See Also](#) | [Example](#)

**Signature**

RetVal = object.Item(Index)

- **object**
  - All Collections The object or objects this property applies to.

- **Index**
  - Variant; input-only
  - The index location in the collection for the member item to query.
  - The index must be either an integer or a string. If an integer, the index must be between 0 and N-1, where N is the number of objects in the collection or selection set.

- **RetVal**
  - Object;
  - The object at the given index location in the collection or selection set.

**Remarks**

This method supports string based iteration. For example, you could reference the document named "siteplan.dwg" through the following
Set mydoc = Docs.Item("siteplan.dwg")

In general, the index can be passed in as any integral type: integer, long, double, etc. In addition, the following collections support a case-sensitive string type as an index (typically the name of the collection member):

- Boundaries
- BreakLines
- ContourItems
- DEMFiles
- DescriptionKeyFile
- DescriptionKeyFiles
- Drawings
- PointGroupNames
- PointFiles
- Projects
- Prototypes
- WaterSheds

Also, the following collections support a case-insensitive string type as an index (typically the name of the collection member):

- Alignments
- Parcels
- PointGroups
- Surfaces

If a station equation is deleted, then the indices for the StationEquations collection will change.
LineIntersection Method

Returns the intersection points of a line with the Alignment.

See Also | Example

Signature

RetVal = object.LineIntersection(StartEasting, StartNorthing, EndEasting, EndNorthing)

object

Alignment The object or objects this property applies to.

StartEasting

Double; input-only
Easting of the first point

StartNorthing

Double; input-only
Northing of the first point

EndEasting

Double; input-only
Easting of the second point

EndNorthing

Double; input-only
Northing of the second point

RetVal
Variant (array of doubles)
The array has the format of: Station, Direction, Easting, Northing.
LoadSetupProfile Method

Load the specified drawing setup profile.

See Also | Example

Signature

object.LoadSetupProfile(ProfileName)

object databasepreferences The object or objects this property applies to.

PropertyName

String; input-only
The profile name to load.

Remarks

The drawing setup files can be found at the path specified by the DrawingSetupPath property of the PreferencesFiles object.
LockPoints Method

Locks a list of point numbers.

See Also | Example

Signature

object.LockPoints(PointString)

object

CogoPoints The object or objects this property applies to.

PointString

String; input-only
Comma delimited string of points to be locked with groups separated by hyphens.

Remarks

Typical examples of the point string format are: "1,2,3,5,10" "1-100,1000-2000"
NewProjectBased Method

Creates a new AutoCAD Land Desktop document.

Signature

RetVal = object.NewProjectBased(TemplateName, DrawingName[, ProjectName])

object

TemplateName
String; input-only
The file name of the template.

DrawingName
String; input-only
The file name for the new drawing.

ProjectName
String; input-only; optional
The name of the Project

RetVal

Document
The newly created Document object.

Remarks

See Also | Example
The drawing name can be up to 255 characters, including path and file extension, and it must be unique. You cannot create two drawings with the same name in the same drawing folder.

It is not necessary to add .dwg to the end of the name; the drawing file extension is created automatically. If a path is not specified, the new drawing will be created at the current DrawingPath for the ActiveProject.

It is not necessary to add .dwt to the end of the TemplateFileName; it will be added automatically if not specified. If the path is not specified, the TemplateDwgPath of the AcadPreferences object will be used.

If a path for the Project is not specified, the Path of the ActiveProject object is used.
OffsetElevationToXy Method (Civil Engineering Feature)

Given an offset and elevation, return the AutoCAD XY coordinates.

See Also | Example

Signature

RetVal = object.OffsetElevationToXy(OffsetElevation)

object

CrossSectionBlock The object or objects this property applies to.

OffsetElevation

Variant (2 element array of doubles); input-only
The offset and elevation to query for.

Retval

Variant (2 element array of doubles); output-only
The XY coordinates for the given offset and elevation.
Open Method

Opens an existing AutoCAD Land Desktop drawing.

**Signature**

object.Open(DrawingName)

object

**Drawings** The object or objects this property applies to.

DrawingName

String; input-only
The name of the drawing to open.

**Remarks**

It is recommended that you save the current drawing before calling this method.

If the drawing is associated with a project name that is different than the **Project**, then you will be prompted to choose a project.

It is not necessary to add .dwg to the end of the name; the drawing file extension will be added automatically if not specified. It is not necessary to specify a path; the current **Path** for the **Drawings** collection will be used if a path is not specified.
OpenProjectBased Method

Opens an existing AutoCAD Land Desktop drawing file (DWG).

See Also | Example

Signature

RetVal = object.OpenProjectBased(DrawingName[, ProjectName])

object

Document The object or objects this property applies to.

DrawingName

String; input-only
The file name for the drawing.

ProjectName

String; input-only; optional
The name of the project

RetVal

Document The newly opened Document object.

Remarks

It is recommended that you save the current drawing before calling this method.

It is not necessary to add .dwg to the end of the name; the drawing file
extension will be added automatically if not specified. If a path is not specified, the path used will be the current DrawingPath for the ActiveProject.

If a path for the Project is not specified, the Path of the ActiveProject object is used.
Paste Method

The Paste method will take another Surface and paste it onto the given Surface.

See Also | Example

Signature

object.Paste(ChildSurface)

object

Surface The object or objects this property applies to.

ChildSurface

String; input-only
Name of the Surface to be added to the given Surface.
PerpIntersection Method

Returns the perpendicular intersection of a point with an Alignment.

See Also | Example

Signature

RetVal = object.PerpIntersection(Easting, Northing)

object

Alignment The object or objects this property applies to.

Easting

Double; input-only
Eastling of the point

Northing

Double; input-only
Northing of the point

RetVal

Variant (array of doubles)
The array has the format of: Station, Direction, Easting, Northing.
PointByNumber Method

Returns a point in the project database by number.

See Also | Example

Signature

RetVal = object.PointByNumber(Number)

object

CogoPoints The object or objects this property applies to.

Number

Long; input-only
The point number to query for.

RetVal

AeccPoint object
The point object.

Remarks

If the point does not exist, then no object will be returned.
PointCodeDescription Method (Civil Engineering Feature)

Returns the description of a cross section point code.

See Also | Example

**Signature**

RetVal = object.PointCodeDescription(Code)

object

**CrossSections** The object or objects this property applies to.

Code

Integer; input-only
The code number to query for.

RetVal

String;
The description of the point code.
PointLocation Method

Given a station and offset, returns the Northing and Eastings.

Signature

object.PointLocation(Station, Offset, Easting, Northing, Direction)

object

Alignment The object or objects this property applies to.

Station

Double; input-only
The Station to query for.

Offset

Double; input-only
The Offset to query for.

Easting

Double; output-only
The Easting of the coordinate.

Northing

Double; output-only
The Northing of the coordinate.

Direction

Double; output-only
The direction of the queried point.

Remarks

If the Offset is to the left of the alignment, it should be negative. If the Offset is to the right of the alignment, it should be positive.
PointNumberFromObjID Method

Given a AutoCAD object ID, PointNumberFromObjID will return the point number.

See Also | Example

Signature

RetVal = object.PointNumberFromObjID(ObjectID)

object

CogoPoints The object or objects this property applies to.

ObjectId

Long; input-only
The AutoCAD entity's object ID. The object is must be a valid AECC_POINT.

RetVal

Long
The point number.

Remarks

The object ID must be for a valid AECC_POINT. If the object is not an AECC_POINT, then the function will return 0 which is an invalid point number.
PointStringToArray Method

Given a point string, returns an array with all of the point numbers.

See Also | Example

Signature

RetVal = object.PointStringToArray(PointString)

object

CogoPoints The object or objects this property applies to.

PointString

String; input-only
Comma delimited string of points to be locked with groups separated by hyphens.

RetVal

Variant (array of longs)
Each long is a point number.

Remarks

Typical examples of the point string format are: "1,2,3,5,10" "1-100,1000-2000"
ProfileByType Method

Returns the Existing or Finished Ground Profile by type.

Signature: Overview

| EGProfiles | (Civil Engineering Feature) |
| FGProfiles | (Civil Engineering Feature) |

Returns the Existing Ground Profile by type.

Signature

RetVal = object.ProfileByType(Type, SurfaceName)

Object

EGProfiles The object or objects this method applies to.

Type

eAeccEGProfileType enum; input-only

RetVal = object.ProfileByType(Type, SurfaceName)

kEgCenter: Existing ground center profile type

kEgLeft: Existing ground left profile type

kEgNone: Existing ground none profile type
kEgNone: Existing ground none profile type

kEgRight: Existing ground right profile type

SurfaceName

String; read-only
The name of the surface.

RetVal

EGProfile object;
The newly added EGProfile object.

Returns the Finished Ground Profile by type.

Signature

RetVal = object.ProfileByType(Type)

Object

FGProfiles
The object or objects this method applies to.

Type

eAeccFGProfileType enum; input-only
Returns the Finished Ground Profile by type.

kFgCenter: Finished ground profile center type

kFgDitchLeft: Finished ground ditch left type

kFgDitchRight: Finished ground ditch right type

kFgLeft1: Finished ground left 1 trans type

kFgLeft2: Finished ground left 2 trans type
RetVal

**FGProfile** object;
The newly added **FGProfile** object.
**RemoveAll Method**

Removes all method for Alignments and StationEquations.

**Signature: Overview**

1. **Alignment**
2. **StationEquations**

Removes all of the entities from the alignment.

**Signature**

object.RemoveAll()

    object  
    **Alignment** The object or objects this property applies to.

**Remarks**

Before changing / adding any entities for an alignment, all entities must be removed.

Removes all station equations from the alignment.

**Signature**
object.RemoveAll()

object

StationEquations
The object or objects this property applies to.
RemoveAllLabels Method

Removes all contour labels.

See Also | Example

Signature

object.RemoveAllLabels()

object

AeccContour The object or objects this property applies to.
RemoveLabelAt Method

Removes the contour label closest to the specified location.

**Signature**

object.RemoveLabelAt(X, Y)

object

* AeccContour The object or objects this property applies to.

**X**

Double; input-only
The X coordinate of the label.

**Y**

Double; input-only
The Y coordinate of the label.

**Remarks**

The label closest to the X, Y point will be removed from the Contour object.
Rename Method

Renames a surface or parcel.

**Signature: Overview**

- **Parcels**
- **Surfaces**

Renames a parcel.

**Signature**

object.Rename(OldName, NewName)

- **object**
  - `Parcels` The object or objects this property applies to.

- **OldName**
  - String; input-only
  - The name of the surface to be renamed.

- **NewName**
  - String; input-only
  - The new name of the parcel.
Renames a surface.

**Signature**

object.Rename(OldName, NewName)

object
- **Surfaces**
  The object or objects this property applies to.

OldName
- String; input-only
  The name of the surface to be renamed.

NewName
- String; input-only
  The new name of the surface.
SampleElevations Method

Given a start and end point, returns an array of derived points from the Surface.

See Also | Example

Signature

RetVal = object.SampleElevations(StartPoint, EndPoint)

object

   Surface  The object or objects this property applies to.

StartPoint

   Variant (3 element array of doubles); input-only
   The starting point to be queried in Easting, Northing, Elevation.

EndPoint

   Variant (3 element array of doubles); input-only
   The ending point to be queried in Easting, Northing, Elevation.

RetVal

   Variant (array of doubles)
   The sample points returned as Easting, Northing, Elevation, Easting, etc.

Remarks
Given a starting and ending point, SampleElevations will return all of the points between the start and end. These points occur when a Surface Edge is crossed. The net result is that the function returns a line that is projected onto the Surface.

SampleElevations will return a Variant. This variant will be an array of doubles. Every 3 doubles is a point of the format Easting, Northing, Elevation.
Save Method

Writes objects to specific databases.

Signature: Overview

- **Alignment**
- **CogoPoint**
- **DescriptionKey**
- **PointGroup**

Saves the Alignment.

**Signature**

```
object.Save()
```

- `object` _Alignment_ The object or objects this property applies to.

Writes the CogoPoint to the project database.

**Signature**

```
object.Save()
```
object

**CogoPoint**
The object or objects this property applies to.

---

 Writes the DescriptionKey to the project database.

**Signature**

object.Save()

object

**DescriptionKey**
The object or objects this property applies to.

---

 Writes the PointGroup to the project database.

**Signature**

object.Save()

object

**PointGroup**
The object or objects this property applies to.
**SaveAsDefault Method**

Saves the DatabasePreferences to the registry.

See Also | Example

**Signature**

object.SaveAsDefault()

object

**DatabasePreferences** The object or objects this property applies to.
SaveSetupProfile Method

Saves the current drawing settings to a drawing setup profile.

Signature

object.SaveSetupProfile(ProfileName)

- object `DatabasePreferences` The object or objects this property applies to.
- ProfileName `String; input-only` The profile name to save to.

Remarks

After you change any of the properties, use the SaveSetupProfile method to save these settings in a drawing setup profile. The new drawing setup profile will be saved at the path specified by the `DrawingSetupPath` property of the `PreferencesFiles` object.
SectionByStation Method (Civil Engineering Feature)

Returns a Cross Section object by specifying its station.

See Also | Example

Signature

RetVal = object.SectionByStation(Station)

object

CrossSections The object or objects this property applies to.

Station

Double; input-only
The station for the desired cross section.

RetVal

CrossSection object
The Cross Section object referenced by the station.
SectionVolume Method (Civil Engineering Feature)

Calculates the cross section volumes.

See Also | Example

Signature

object.SectionVolume(Type, CurveCorrection, Swell, Shrink, CutArea, CutCenter, CutVolume, FillArea, FillCentroid, FillVolume)

object

_CrossSection_ The object or objects this property applies to.

Type

eAeccCrossSectionVolumeMethod enum; input-only.
Specifies the setting to return.

kVolumeByAvgEndArea: Volume by average end area calculation

kVolumeByPrismodal Volume by prismatic calculation

CurveCorrection

Double; input-only
Applies a correction factor to the volume equal to the ratio of the centroid to the radius of the curve. 1 - curve correction on
0 - curve correction off

**Swell**

Double; input-only
The swell factor

**Shrink**

Double; input-only
The shrink factor

**CutArea**

Double; output-only
The cut area

**CutCentroid**

Double; output-only
The cut centroid

**CutVolume**

Double; output-only
The cut volume

**FillArea**

Double; output-only
The fill area

**FillCentroid**

Double; output-only
The fill centroid

**FillVolume**

Double; output-only
The fill volume
SetBoundingBox Method

Sets the bounding box for the surface.

See Also | Example

Signature

object.SetBoundingBox(LowerLeftPoint, UpperRightPoint)

object

Surface The object or objects this property applies to.

LowerLeftPoint

Variant (3 element array of doubles); input-only
The lower left coordinate for the bounding box in Easting, Northing, Elevation format.

UpperRightPoint

Variant (3 element array of doubles); input-only
The upper right coordinate for the bounding box in Easting, Northing, Elevation format.

Remarks

Calculations will be much faster if a bounding box is set. If no bounding box is set for a surface, then one will be generated when the surface is built.
SetDouble Method

Sets the specified preference setting.

See Also | Example

Signature

object.SetDouble(Setting, Value)

object

PreferencesAlignment,
PreferencesCrossSection, (Civil Engineering Feature)
PreferencesCogo,
PreferencesProfile, (Civil Engineering Feature)
PreferencesSurface
The object or objects this property applies to.

Setting

eAeccPrefAlignDouble,
eAeccPrefCogoDouble,
eAeccPrefCrossSectionDouble,
eAeccPrefProfileDouble,
eAeccPrefSurfaceDouble
enum; input-only
Specifies the setting to set.

Value

Double; input-only
The value of the setting.
Remarks

For a listing of settings see the `GetDouble` method.
SetInteger Method

Sets the specified preference setting.

See Also | Example

**Signature**

```
object.SetInteger(Setting, Value)
```

- **object**
  - `PreferencesAlignment`, `PreferencesCogo`, `PreferencesCrossSection`, (Civil Engineering Feature)
  - `PreferencesParcel`, `PreferencesProfile`, (Civil Engineering Feature)
  - `PreferencesSurface`
  - The object or objects this property applies to.

- **Setting**
  - `eAeccPrefAlignInt`, `eAeccPrefCogoInt`, `eAeccPrefCrossSectionInt`, `eAeccPrefParcelInt`, `eAeccPrefProfileInt`, `eAeccPrefSurfaceInt`, `enum; input-only`
  - Specifies the setting to set.

- **Value**
  - `Integer; input-only`
The value of the setting.

Remarks

For a listing of settings see the `GetInteger` method.
SetReferenceCurve Method

Sets the reference curve for the Curve Text.

See Also | Example

Signature

object.SetReferenceCurve(CircleOrArcEntity)

object

*AeccCurveText* The object or objects this property applies to.

CircleOrArcEntity

*AcadEntity object; input-only*

A circle or arc entity.
**SetString Method**

Sets the specified preference setting.

**See Also | Example**

**Signature**

object.SetString(Setting, Value)

object

PreferencesAlignment, PreferencesCogo, PreferencesCrossSection, (Civil Engineering Features) PreferencesParcel, PreferencesProfile, (Civil Engineering Features) PreferencesSurface,

The object or objects this property applies to.

**Setting**

eAeccPrefAlignString, eAeccPrefCogoString, eAeccPrefCrossSectionString, eAeccPrefParcelString eAeccPrefProfileString, eAeccPrefSurfaceString, enum; input-only

Specifies the setting to set.

**Value**

String; input-only
The value of the setting.

Remarks

For a listing of settings see the GetString method.
StationElevationToXy Method (Civil Engineering Features)

Given a station and elevation, return the AutoCAD XY coordinates.

See Also | Example

Signature

RetVal = object.StationElevationToXy(StaElev)

object

(ProfileBlock) The object or objects this property applies to.

StaElev

Variant (2 element array doubles); input-only
The Station and Elevation to query for.

RetVal

Variant (2 element array doubles); output-only
The XY coordinates for the given station and elevation.
StationOffset Method

Given a Northings and Eastings, returns the station, an offset, and the polar direction.

See Also | Example

Signature

object.StationOffset(Easting, Northing, Station, Offset, Direction)

object

Alignment Double; input-only
The object or objects this property applies to.

Easting

Double; input-only
The Easting of the coordinate to query for.

Northing

Double; input-only
The Northing of the coordinate to query for.

Station

Double; output-only
The Station of the queried point.

Offset

Double; output-only
The Offset of the queried point.

Direction
Double; output-only
The direction of the queried point.

Remarks

If the Offset is to the left of the alignment, it will be negative. If the Offset is to the right of the alignment, it will be positive.
UnlockPoints Method

Unlocks a list of point numbers.

See Also | Example

**Signature**

object.UnlockPoints(PointString)

- **object**: `CogoPoints` The object or objects this property applies to.
  - **PointString**: String; input-only
    - Comma delimited string of points to be unlocked with groups separated by hyphens.

**Remarks**

Typical examples of the point string format are: "1,2,3,5,10" "1-100,1000-2000"
**XyToEastNorth Method**

Given a AutoCAD X, Y, returns the Easting and Northing.

**Signature**

RetVal = object.XyToEastNorth(Xy)

- **object**
  - Utility
  - **Utility** The object or objects this property applies to.

- **Xy**
  - Variant (3 element array of doubles); input-only
  - The X and Y coordinates.

- **RetVal**
  - Variant (3 element array of doubles)
  - The coordinates in Easting and Northing.

**Remarks**

EastNorth must be declared as a variant and not as an array. To access the Easting and Northing, subscript the variant ( var(0), var(1)). The third element (elevation) is unchanged.
**XyToOffsetElevation Method (Civil Engineering Features)**

Given an AutoCAD XY coordinates, return the cross section offset and elevation.

**Signature**

RetVal = object.XyToOffsetElevation(Xy)

- **object**
  - CrossSectionBlock
    - The object or objects this property applies to.

- **Xy**
  - Variant (2 element array of doubles); input-only
    - The X and Y coordinates to query for.

- **RetVal**
  - Variant (2 element array of doubles); output-only
    - The offset and elevation for the given XY coordinates.
XyToStationElevation Method (Civil Engineering Feature)

Given an AutoCAD XY, return the profile station and elevation.

See Also | Example

Signature

RetVal = object.XyToStationElevation(Xy)

object

ProfileBlock The object or objects this property applies to.

Xy

Variant (2 element array doubles); input-only
The X and Y coordinates to query for.

RetVal

Variant (2 element array doubles); output-only
The station and elevation for the given XY coordinates.
Properties

A The spiral's A value.

Active Specifies if the projects is currently loaded.

ActiveDocument Specifies the active document (drawing file).

ActiveProject Specifies the currently opened Land Desktop project

AD1 Returns AD1 which is the radius at the end of the spiral (SC).

AD2 Returns AD2 which is the radius at the start of the spiral (TS).

AlignEntities The alignment's entities collection.

Alignment The collection of alignment preferences for the project.

Alignments Gets the Alignments collection.

AngularAzimuth The format used to display angles.

AngularDisplayFormat The units used to display angles.

AngularPrecision Specifies the angular precision for the drawing.

Application Gets the AeccApplication object.

Area The area of the parcel.

Area2D The 2D area of a Surface or Face.

Area3D The 3D area of a Surface or Face.

AreaDisplayUnit The units used to display areas.
**AreaPrecision** The precision used to display areas.

**AreaSuffix** The suffix displayed after an area.

**Application** Gets the AeccApplication object.

**Author** Specifies the author of the project.

**AutoSave** Returns and sets the AutoSave for Alignments, CogoPoints, DescriptionKeyFiles, and PointGroups.

**AverageGrade** Returns the BreakLines collection.

**B**

**BasePoint** Gets and sets the base point for the drawing as an XYZ.

**BasePointNE** Gets and sets the base point for the drawing as an ENZ.

**BeginCondition** Returns the BeginCondition for the Spiral.

**BorderBlockFileName** Specifies the block filename to insert as a border.

**BorderBottomMargin** Specifies the bottom margin for the border.

**BorderLeftMargin** Specifies the left margin for the border.

**BorderLineWidth** Specifies the line width of the border.

**BorderPath** Specifies the directory in which the border files are located.

**BorderRightMargin** Specifies the right margin for the border.

**BorderStyle** Specifies the border style to use for the drawing.

**BorderTopMargin** Specifies the top margin for the border.

**Boundaries** Returns the Boundaries collection.

**Boundary** Returns the boundary of the Watershed.
**BreakLines** Returns the BreakLines collection.

**CCWFlag** The counter-clockwise flag for the curve or parcel.

**CenterEastings** The Easting coordinate for the alignment curves or parcel curves center.

**CenterlineOffset** (Civil Engineering Feature) Returns the centerline offset of the cross section.

**CenterNorthing** The Northing coordinate for the alignment curves or parcel curves center.

**CentroidEastings** The easting coordinate of the parcel centroid.

**CentroidNorthing** The northing coordinate of the parcel centroid.

**ChordDirection** The direction of the chord for the curve.

**ChordLength** The chord length of the alignment or parcel curve.

**CivilDataFilesPath** Specifies the directory in which the Civil Engineering Feature data is located.

**Code** Returns the cross section point code or description key code.

**Cogo** The collection of cogo preferences for the project.

**CogoPoints** Gets the CogoPoints collection.

**ContourItems** Returns the ContourItems collection.

**ContourStyle** Returns and set the Contour Style object for this contour.

**ContourStyleName** Returns the Contour Style name.

**ContourStylesPath** Specifies the directory in which the contour styles are located.
**CoordinatePrecision** Specifies the coordinate precision for the drawing.

**Coordinates** The coordinate properties for various objects.

**CoordinateZone** Specifies the coordinate zone.

**Count** Gets the number of items in the collection.

**Course** Returns the Course of the parcel entity.

**CourseIn** Returns the Course In of the Parcel.

**CourseOut** Returns the Course Out of the Parcel.

**CrossSection** (Civil Engineering Feature) The collection of cross section preferences for the project.

**CrossSectionBlocks** (Civil Engineering Feature) Gets the CrossSectionBlock object.

**CrossSectionPointCodes** (Civil Engineering Feature) Returns the cross sections point code collection.

**CrossSections** (Civil Engineering Feature) Returns the alignment cross sections collection.

**CrossSectionSurfaces** (Civil Engineering Feature) Returns the cross sections surfaces collection.

**CurrentAlignment** Sets and gets the current alignment.

**CurrentSurface** Gets / sets the current Surface.

**CurveCode** (Civil Engineering Feature) Returns the superelevation curve code.

**CurveLabelIncrement** (Civil Engineering Feature) Returns the distance between vertical curve elevation labels.

**CurveLength** (Civil Engineering Feature) Returns the Curve Length of the PVI.
**DatabaseScale** Specifies the database scale at which the horizontal axis of the drawing is displayed.

**DataPath** Specifies the directory in which the Land Desktop data files are installed.

**DatumElevation** (Civil Engineering Feature) Returns the datum elevation for the alignment profile or cross section.

**Delta** The delta value for a AlignCurve, AlignSpiral, or ParcelCurve.

**DEMFiles** Returns the DEMFiles collection.

**Description** The description properties for various objects.

**DescriptionFormat** Specifies the description key format.

**DescriptionKeyFiles** Gets the DescriptionKeyFiles collection.

**DescriptionLayer** Specifies the description key layer.

**DescriptionOverride** Specifies if the description for the project database point will be overridden from the Point Group.

**DescriptionXDRef** Specifies the description override for the project database point group.

**Direction** The Direction of the ProfileBlock or alignment tangent.

**Documents** Gets the Document collection.

**DrainsInto** Returns an array of Ids that the Watershed drains into.

**DrawingPath** Specifies the drawing path of the project.

**Drawings** Gets the Drawings collection.

**DrawingSetupPath** Specifies the directory in which the drawing setup files are located.
**E**

**Easting** The Easting coordinates properties for various objects.

**Edges** Returns the Edges collection.

**EGPrecision** (Civil Engineering Feature) Returns the existing ground precision for the alignment profile or cross section.

**EGProfiles** (Civil Engineering Feature) Gets the existing ground profiles collection.

**Elevation** The elevation property for various objects.

**ElevationContours** Returns the ElevationContours collection.

**ElevationOverride** Specifies if the elevation for the project database point will be overridden from the Point Group.

**ElevationPrecision** Specifies the elevation precision for the drawing.

**ElevationXDRef** Specifies the elevation override for the project database point group.

**EndDirection** The ending direction of the curve.

**EndEasting** The Easting coordinate for the end of the individual object entities.

**EndingStation** The object's ending station.

**EndNorthing** The Northing coordinate for the end of the individual object entity.

**ExceedBoth** If true, then both minimum depression settings must be met.

**ExtEasting** The Easting coordinate for the External point (Ext).

**ExternalSecant** The external secant length for the curve.

**ExtNorthing** The Northing coordinate for the External point (Ext).
Faces Returns the Faces collection.

FacetDeviation Gets and sets the facet deviation.

FGPrecision (Civil Engineering Feature) Returns the finished ground precision for the alignment profile or cross section.

FGProfiles (Civil Engineering Feature) Gets the FGProfiles collection.

File The filename of the file lock.

FileLocks Gets the FileLocks collection.

Files Gets the PreferencesFiles object.

FirstTimeDrawingSetup Selects how new drawings are setup.

FirstTimeDrawingSetupFile The drawing setup file loaded automatically for a new drawing.

Format The format for the PointFile.

FormatsPath Specifies the directory in which the point import and export formats are located.

FullDescription Returns the full description for the Cogo Point.

FullName The full name includes the path and file name.

GridEastings The grid Easting for the COGO project database point.

GridNorthing The grid Northing for the COGO project database point.

GroupName Gets / sets the group name for various objects.
**Height** (Civil Engineering Feature) Returns the height of the alignment cross section.

**HelpPath** Specifies the directory in which the help files are installed.

**Id** The identification number for the Boundary, BreakLine, ContourItem or Watershed.

**Inputs** Returns the SurfaceInputs object for the surface.

**IsBreakLine** Is the Boundary considered a BreakLine?

**IsNameSupported** Returns whether point names are supported or not.

**IsVisible** Returns if the Face is visible.

**J**

**K**

**K** The spiral's K value.

**Keywords** Specifies the keywords for the project.

**L**

**Label** The name of the locked data file or folder.

**LabelPoints** Returns the contour label points.

**LabelStyle** Specifies the label style override for the project database point group.

**LabelStyleOverride** Specifies if the label style for the project database point will be overridden from the Point Group.

**LabelStylePath** Specifies the directory in which the line, curve, spiral and point label style files are located.
**LabelStyleXDRRef** Specifies the label style override for the project database point group.

**LastUsedDwg** The last used drawing.

**LastUsedDwgPath** The last used drawing path.

**LastUsedProj** The last used project.

**LastUsedProjPath** The last used project path.

**Latitude** The latitude of the point.

**LayerFile** The layer file to apply to the drawing.

**LayerStandard** The layer standard to apply to the drawing.

**LeftWidth** (Civil Engineering Feature) Returns the left width of the alignment cross section.

**Length** The length of the alignment curve, spiral or tangent or parcel curve or line.

**LExt** Returns the external length of the spiral.

**LinearDisplayUnit** The format used to display linear values.

**LinearPrecision** Specifies the linear precision for the drawing.

**LinearUnit** The unit used to display linear values.

**LockedPointNumbers** Returns a list of the locked points in the current project point database.

**LockType** Specifies the type of lock on the Alignment, COGO Point, or Surface.

**LOffset** The offset distance for the spiral.

**Longitude** The longitude of the point.
MaskBlockStyles Gets the Masking Block Styles collection for the drawing.

MassGroups Gets the MassGroups collection for the drawing.

MaxElevation Returns the Maximum Elevation of a Cross Section or Surface.

MaxFaceArea The area of the largest Face for the Surface.

MaxGrade The maximum grade for the Surface.

MaxOffset (Civil Engineering Feature) Returns the Maximum Offset of the Cross Section.

MeanElevation The mean elevation of the Surface.

MeausurementUnit The system of units used to display measurements.

MidOrdinate The mid ordinate for the curve.

MinDepressionArea The minimum depression area for all watersheds.

MinDepressionDepth The minimum depression depth for all watersheds.

MinElevation Returns the Minimum Elevation of a Cross Section or Surface.

MinFaceArea The area of the smallest Face for the Surface.

MinGrade The minimum grade for the Surface.

MinOffset (Civil Engineering Feature) Returns the Minimum Offset of the Cross Section.

MVBlockStyles Gets the Multiview Block Styles collection for the drawing.

N
**Name** Gets the name of various objects.

**NameOverride** Specifies if the name for the project database point will be overridden from the Point Group.

**NameXDRef** Specifies the Name override for the project database point group. The name will be taken from the external database reference (XDRef).

**NextPointNumber** Gets / sets the next sequential point number for point creation.

**Normal** Returns the Normal for the Face.

**Northing** Gets the Northing coordinates of various objects.

**NorthRotation** Specifies the north rotation for your drawing layout.

**Number** The number property for various objects.

**NumberOfFaces** The number of Faces for the Surface.

**NumberOfPoints** The number of TinPoints for the Surface.

**ObjectID** Returns the Object ID for the entity.

**Offset** (Civil Engineering Feature) Returns the cross section point code offset.

**OffsetElevations** (Civil Engineering Feature) Returns the cross section surface offset and elevations.

**Outputs** Returns the SurfaceOutputs object for the surface.

**OverflowPoints** Returns the WaterShed overflow points.

**OverrideDescription** Returns the description that has been overridden by the PointGroup.
OverrideElevation Returns the elevation that has been overridden by the PointGroup.

OverrideName Returns the point name that has been overridden by the PointGroup.

OverrideNew Toggles the display of the Land Desktop New dialog box.

OverrideOpen Toggles the display of the Land Desktop Open dialog box.

Owner The owner (AutoCAD login name) of the lock owner.

P

P The spiral's P value.

Parcel The collection of parcel preferences for the project.

ParcelEntities The collection of geometric entities for the Parcel.

Parcels Gets the Parcels collection.

Path Gets the path of various objects.

Perimeter The perimeter of the parcel.

PiEasting The Easting coordinate for the curve's PI.

PiNorthing The Northing coordinate for the curve's PI.

PointFiles Returns the point files collection.

PointGroupNames Returns the PointGroupNames collection.

PointGroups Gets the PointGroups collection.

PointList Specifies the list of project database points for the point group.

PointNameSize Returns maximum number of characters supported in point name.
**PointOnlineTolerance** Returns the minimum distance between a line and a point not on the line.

**PointTolerance** Returns the minimum distance between distinct points.

**Precision** The precision used to truncate area labels when a Parcel is imported to the drawing.

**Preferences** The preferences property of various objects.

**PreferencesPath** Specifies the directory in which the preference settings are located.

**Profile** Gets the vertical profile settings for the project.

**ProfileBlocks** (Civil Engineering Feature) Gets the ProfileBlocks object.

**ProfileStyles** Gets the ProfileStyles collection for the drawing.

**ProgramPath** Specifies the directory in which the Land Desktop is installed.

**ProjectName** The name of the project that the drawing belongs to.

**ProjectPath** Gets the project path of various objects.

**Projects** Gets the Projects collection.

**PrototypeName** Specifies the name of the prototype used for the project.

**PrototypePath** Specifies the directory in which the project prototype files are located.

**Prototypes** Gets the Prototypes collection.

**PVIs** Gets the PVIs collection.

Q

R
**RadialDistance** The radial distance of the spiral.

**Radius** The radius of the alignment or parcel curve.

**RawDescription** The raw description for the COGO project database point.

**RevisionNumber** The revision number for the Surface.

**RightWidth** (Civil Engineering Feature) Returns the right width of the alignment cross section.

**RotateByDescriptionParam** If TRUE, only the RotateDescriptionParam is rotated.

**RotateByFixedFactor** If TRUE, rotate by the RotateFactor.

**RotateClockwise** The rotation direction.

**RotateDescriptionParam** The description parameter for rotation.

**RotateFixedFactor** The rotation factor.

**S**

**ScaleByDescriptionParam** If TRUE, only the ScaleDescriptionParam is scaled.

**ScaleByDrawingScale** If TRUE, the drawing scale factor is applied.

**ScaleByFixedFactor** If TRUE, the fixed scale factor ScaleFactor is applied.

**ScaleDescriptionParam** The description parameter for scaling.

**ScaleFixedFactor** The fixed scale factor.

**ScaleInXY** If TRUE, the symbol is scaled in the XY plane

**ScaleInZ** If TRUE, the symbol is scaled in the Z axis.
**ScaleOnInsert** Specifies whether objects are automatically scaled when inserted into the drawing.

**SearchType** Returns how the Faces collection was generated.

**SheetHeight** Specifies the height of the sheet.

**SheetWidth** Specifies the width of the sheet.

**ShortTangent** Returns the short tangent length for compound and simple spirals.

**ShowStartupDialog** Toggles the display of the startup dialog.

**ShowSubfolders** Determines whether drawings in subfolders are included.

**SpeedTablesPath** Specifies the directory in which the speed tables are located.

**SpiEasting** The Easting coordinate for the spi point.

**SpiTangent** Returns the long tangent length.

**SpiNorthing** The Northing coordinate for the spi point.

**SpiralType1** Returns the calculation method of the spiral.

**SpiralType2** Returns whether the spiral is a simple, offset, or compound spiral.

**StartDirection** The starting direction of the curve or spiral.

**StartEasting** The Easting coordinate for the beginning of the individual alignment or profile entities.

**StartingStation** The object's starting station.

**StartNorthing** The Northing coordinate for the beginning of the individual alignment or profile entities.
\textbf{Station} (Civil Engineering Feature) Returns the station for the cross section, cross section block, superelevation, or PVI.

\textbf{StationAhead} The ahead station for the station equation.

\textbf{StationBack} The back station for the station equation.

\textbf{StationElevations} (Civil Engineering Feature) An array of stations and elevations for the VerticalSurface

\textbf{StationEquations} The alignment's station equations collection.

\textbf{StationIncrement} (Civil Engineering Feature) Returns the horizontal distance between vertical grid lines.

\textbf{Status} The status for a surface.

\textbf{SuperelevationCode} (Civil Engineering Feature) Returns the superelevation code.

\textbf{Surface} The collection of surface preferences for the project.

\textbf{SurfaceName} (Civil Engineering Feature) The name of the EGProfile surface.

\textbf{Surfaces} Gets the Surfaces collection.

\textbf{SymbolBlock} Specifies the symbol block.

\textbf{SymbolLayer} Specifies the symbol block layer.

\textbf{SymbolManagerPath} Specifies the directory in which the symbol sets are located.

\textbf{SystemPath} Specifies the directory in which the Land Desktop is installed.

\textbf{T}

\textbf{TangentLabelIncrement} (Civil Engineering Feature) Returns the distance between tangent elevation labels.
**TangentLength** The tangent length of the curve.

**TempPath** Specifies the directory in which the temporary files are stored.

**TextAbove** The text above the object.

**TextBelow** The text below the object.

**TextHeight** The height of the text in the drawing.

**TextOffsetAbove** The offset of the text above the object.

**TextOffsetBelow** The offset of the text below the object.

**TextSize** The size of the text above and below the object.

**ThetaExt** Returns the external theta for a compound spiral.

**Time** The lock creation date and time.

**TinPoints** Returns the TinPoints collection.

**TotalX** The total X value for the spiral.

**TotalY** The total Y value for the spiral.

**Type** The type properties for Objects.

**U**

**UpperRight** Returns the Upper Right Coordinates of the ProfileBlock as XY.

**UsedPointNumbers** Returns the list of used point numbers in the current project database.

**User** Gets the PreferencesUser object.

**Utility** Gets the Utility object.

**V**
**VerticalScale** Specifies the scale at which the vertical axis of the drawing is displayed, or the vertical scale for alignment profiles and cross sections.

**Volume** The volume of the surface.

**VolumeDisplayUnit** The units used to display volumes.

**VolumePrecision** The precision used to display volumes.

**VolumeSuffix** The suffix displayed after an volume.

W

**WaterSheds** Returns the WaterSheds collection.

X

Y

Z
A Property

The spiral's A value.

See Also | Example

Signature

object.A

object

AlignSpiral The object or objects this property applies to.

A

Double; read-only

The spiral's A value.
Active Property

Specifies if the projects is currently loaded.

Signature

object.Active

object

Project The object or objects this property applies to.

Active

Boolean; read-only
TRUE: The project is loaded.
FALSE: The project is not loaded.

Remarks

The ActiveProject property of the AeccApplication object is the only instance of AeccProject with this property set to TRUE. This indicates that access to project owned data in external databases, such as CogoPoints, is possible.
ActiveDocument Property

Specifies the active document (drawing file).

See Also | Example

Signature

object.ActiveDocument

object

AeccApplication The object or objects this property applies to.

ActiveDocument

Document object; read-only

Remarks

In the Preview, the ActiveDocument property returns the opened document (an AutoCAD Land Desktop drawing).
**ActiveProject Property**

Specifies the currently opened AutoCAD Land Desktop project.

**Signature**

`object.ActiveProject`

`object`  
`AeccApplication`  The object or objects this property applies to.

`ActiveProject`  
`Project` object; read-only

**Remarks**

The `ActiveProject` property allows access to all project data.
AD1 Property

Returns AD1 which is the radius at the end of the spiral (SC).

See Also | Example

Signature

object.AD1

object

AlignSpiral The object or objects this property applies to.

AD1

Double; read-only
The radius at the end of the spiral (SC).
AD2 Property

Returns AD2 which is the radius at the start of the spiral (TS).

See Also | Example

**Signature**

object.AD2

object

   **AlignSpiral** The object or objects this property applies to.

AD2

   Double; read-only
   The radius at the start of the spiral (TS).

**Remarks**

AD2 is the radius at the start of the spiral (TS = -99999.99 for infinite radius, or a value for compound spiral).
AlignEntities Property

The alignment's entities collection.

See Also | Example

Signature

object.AlignEntities

object

Alignment The object or objects this property applies to.

AlignEntities

AlignEntities collection; read-only
The AlignEntities collection.
Alignment Property

The collection of alignment preferences for the project.

See Also | Example

Signature

object.Alignment

object

object

PreferencesProject The object or objects this property applies to.

Alignment

PreferencesAlignment; read-only

The PreferencesAlignment object.
## Alignments Property

Gets the Alignments collection.

### Signature

`object.Alignments`

- `object`  
  - `Project`  
    - The object or objects this property applies to.

- `Alignments`  
  - `Alignments` collection; read-only
  - The `Alignments` collection.

### Remarks

The `Alignments` collection represents all of the `Alignment` objects in this project. This property is valid only for the `ActiveProject`.
AngularAzimuth Property

The format used to display angles.

See Also | Example

Signature

object.AngularAzimuth

object

DatabasePreferences The object or objects this property applies to.

AngularAzimuth

Long; read-write

0: Use Bearings

1: Use North Azimuths

2: Use South Azimuths
AngularDisplayFormat Property

The units used to display angles.

See Also | Example

Signature

object.AngularDisplayFormat

object

DatabasePreferences The object or objects this property applies to.

AngularDisplayFormat

Long; read-write

1: Use Degrees/Minutes/Seconds

2: Use Grads

Remarks

When you type degrees at AutoCAD Land Desktop command prompts, type them in the decimal format indicated (DD.MMSS). Use a period (.) between the degree value and the minutes and seconds. For example, to enter 67°45'15" type the value as 67.4515. If you use bearings, then type the bearing quadrant first, and then the angle in degrees.
AngularPrecision Property

Specifies the angular precision for the drawing.

**Signature**

object.AngularPrecision

object

[DatabasePreferences](#) The object or objects this property applies to.

AngularPrecision

Long; read-write

The number of decimals to the right of the decimal point.

**Remarks**

This is used to display minutes and seconds.

The precision settings are used only for labeling and listing values, not actual computations. AutoCAD Land Desktop commands always calculate all numbers up to the highest internal precision.
Application Property

Gets the AeccApplication object.

See Also | Example

Signature

object.Application

object

All objects The object or objects this property applies to.

Application

Application object, read-only

Remarks

The Application object represents the AutoCAD Land Desktop, and provides the means to navigate to the top of the object hierarchy. All objects in the object model implement the Application property.
Area Property

The area of the parcel.

See Also | Example

Signature

object.Area

object

Parcel The object or objects this property applies to.

Area

Double; read-only
The area of the parcel.
Area2D Property

The 2D area of a Surface or Face.

Signature: Overview

<table>
<thead>
<tr>
<th>Face</th>
</tr>
</thead>
<tbody>
<tr>
<td>Surface</td>
</tr>
</tbody>
</table>

The 2D area of a Face.

Signature

object.Area2D

object

Face The object or objects this property applies to.

Area2D

Double; read-only

The 2D face area.

Remarks

The 2D face area is the apparent triangle area if you look at the face from plan view.

The 2D area of a Surface.
**Signature**

`object.Area2D`

`object`

*Surface*

The object or objects this property applies to.

`Area2D`

Double; read-only

The 2D surface area.

**Remarks**

The 2D surface area is the apparent surface area if you look at the surface from plan view. It is obtained by projecting the visible triangles onto the XY plane along the Z axis and summing the areas of the triangles.

If areas on the surface are hidden within boundaries, then these areas are not included in the surface area.

This is an Extended Statistic.
Area3D Property

The 3D area of a Surface or Face.

Signature: Overview

| Face |

| Surface |

The 3D area of the Face.

Signature

object.Area3D

object

Face The object or objects this property applies to.

Area3D

Double; read-only
The 3D face area.

Remarks

The 3D face area is the true area of the triangle.

The 3D area of the Surface.
Signature

object.Area3D

object

Surface
The object or objects this property applies to.

Area3D

Double; read-only
The 3D surface area.

Remarks

The 3D surface area is the true area of the surface and accounts for variations in the surface elevation. The 3D area is the sum of the areas of each of the visible triangles in the surface without projecting the triangles. The greater the variation in elevations, the more the 3D area will differ from the 2D area.

An example of the difference between 2D and 3D areas is a building pad. The 2D area is the area that you can see from only the top and doesn't vary with the height of the building pad. The 3D surface area of the building pad is the sum of the areas of each face of the building pad, which would become larger the higher the building pad becomes.

This is an Extended Statistic.
AreaDisplayUnit Property

The units used to display areas.

See Also | Example

**Signature**

object.AreaDisplayUnit

- object
  - [Database_preferences](#) The object or objects this property applies to.

- AreaDisplayUnit
  - AecBuiltInUnit enum; read-write
    - aecUnitSquareInch: Square inch unit
    - aecUnitSquareFoot: Square foot unit
    - aecUnitSquareYard: Square yard unit
    - aecUnitSquareMil: Square millimeter unit
    - aecUnitSquareCentimeter: Square centimeter unit
    - aecUnitSquareDecimeter: Square decimeter unit
    - aecUnitSquareMeters: Square meter unit
Remarks

The AreaDisplayUnit supports more units than are shown above. The units shown above are those that are supported by the AutoCAD Land Desktop.
AreaPrecision Property

The precision used to display areas.

See Also | Example

Signature

object.AreaDisplayUnit

object

DatabasePreferences The object or objects this property applies to.

AreaPrecision

Long; read-write
The display precision used for areas.

Remarks

The precision settings are used only for labeling and listing values, not actual computations. AutoCAD Land Desktop commands always calculate all numbers up to the highest internal precision.
**AreaSuffix Property**

The suffix displayed after an area.

**Signature**

object.AreaDisplayUnit

object  

DatabasePreferences The object or objects this property applies to.

AreaPrecision  

String; read-write  
The suffix displayed after an area.

**Remarks**

The default value is "Sq.Ft." or "m2", depending on the value of the MeasurementUnit property. The value must 255 characters or less in length.
Author Property

Specifies the author of the project.

See Also | Example

Signature

object.Author

    object

    Project The object or objects this property applies to.

    Author

    String; read-write
    The author of the project.
AutoSave Property

Returns and sets the AutoSave for Alignments, CogoPoints, DescriptionKeyFiles, and PointGroups.

See Also | Example

**Signature: Overview**

- Alignments
- CogoPoints
- DescriptionKeyFiles
- PointGroups

Returns and sets the AutoSave.

**Signature**

object.AutoSave

object

Alignments The object or objects this property applies to.

AutoSave

Boolean; read-write
TRUE: AutoSave is enabled
FALSE: AutoSave is disabled
Remarks

The initial value for this property is TRUE. Each addition or change is written to the Alignments file immediately. If this property is set to FALSE, the Save method must be called to write additions or changes to the Alignment file. The Add method for Alignments is always executed immediately, regardless of the AutoSave setting.

Returns and sets the AutoSave.

Signature

object.AutoSave

    object

    CogoPoints
    The object or objects this property applies to.

    AutoSave
    Boolean; read-write
    TRUE: AutoSave is enabled
    FALSE: AutoSave is disabled

Remarks

The initial value for this property is TRUE. Each addition or change is written to the CogoPoints file immediately. If this property is set to FALSE, the Save method must be called to write additions or changes to the CogoPoints file. The Add method for CogoPoints is always executed immediately, regardless of the AutoSave setting.

Returns and sets the AutoSave.

Signature

object.AutoSave

    object
**DescriptionKeyFiles**
The object or objects this property applies to.

**AutoSave**
Boolean; read-write
TRUE: AutoSave is enabled
FALSE: AutoSave is disabled

**Remarks**
The initial value for this property is TRUE. Each addition or change is written to the DescriptionKeyFiles file immediately. If this property is set to FALSE, the `Save` method must be called to write additions or changes to the DescriptionKeyFiles file. The `Add` method for `DescriptionKeyFiles` and `DescriptionKeyFile` are always executed immediately, regardless of the AutoSave setting.

Returns and sets the AutoSave.

**Signature**

```csharp
object.AutoSave
```

**PointGroups**
The object or objects this property applies to.

**AutoSave**
Boolean; read-write
TRUE: AutoSave is enabled
FALSE: AutoSave is disabled

**Remarks**
The initial value for this property is TRUE. Each addition or change is written to the PointGroups collection immediately. If this property is set to FALSE, the `Save` method must be called to write additions or changes to the PointGroups collection. The `Add` method for `PointGroups` is always
executed immediately, regardless of the AutoSave setting.
AverageGrade Property

The average grade for the Surface.

See Also | Example

Signature

object.AverageGrade

  object

  Surface The object or objects this property applies to.

  AverageGrade

    Double; read-only
    The average grade for the surface.

Remarks

This is an Extended Statistic.
BasePoint Property

Gets and sets the base point for the drawing as an XY.

See Also | Example

**Signature**

object.BasePoint

object

(DatabasePreferences) The object or objects this property applies to.

BasePoint

Variant (3 element array of doubles); read-write

The base point in WCS X, Y, Z.

**Remarks**

Set this value to indicate the WCS X, Y, Z that corresponds to the value of the BasePointNE property. In the default coordinate system X,Y is 0,0 and the Northing/Easting is 0,0.
BasePointNE Property

Gets and sets the base point for the drawing as an ENZ.

See Also | Example

**Signature**

object.BasePointNE

object

DatabasePreferences The object or objects this property applies to.

BasePointNE

Variant (3 element array of doubles); read-write
The base point in WCS X, Y, Z.

**Remarks**

Set this value to indicate the Easting, Northing and elevation that corresponds to the value of the BasePoint property. In the default coordinate system X,Y is 0,0 and the Northing/Easting is 0,0.
BeginCondition Property

Returns the BeginCondition for the Spiral.

See Also | Example

**Signature**

object.BeginCondition

  object

    **AlignSpiral** The object or objects this property applies to.

BeginCondition

  eAeccSpiralBegin enum; read-only

  kSC:    Spiral begins on the SC point

  kTS:    Spiral begins on the TS point

**Remarks**

The BeginCondition determines whether the spiral begins on its SC or TS point.
BorderBlockFilename Property

Specifies the block filename to insert as a border.

See Also | Example

Signature

object.BorderBlockFilename

object

DatabasePreferences The object or objects this property applies to.

BorderBlockFilename

String; read-write
The border block filename.

Remarks

The block names comply with the following naming conventions:

- pf_ are plan borders for drawings that use feet as units
- pm_ are plan borders for drawings that use meters as units
- df_ are detail borders for drawings that use feet as units
- dm_ are detail borders for drawings that use meters as units
BorderBottomMargin Property

Specifies the bottom margin for the border.

See Also | Example

Signature

object.BorderBottomMargin

object

DatabasePreferences The object or objects this property applies to.

BorderBottomMargin

Double; read-write
The border bottom margin.
BorderLeftMargin Property

Specifies the left margin for the border.

Signature

object.BorderLeftMargin

object

DatabasePreferences The object or objects this property applies to.

BorderLeftMargin

Double; read-write
The border left margin.
BorderLineWidth Property

Specifies the line width of the border.

See Also | Example

Signature

object.BorderLineWidth

object

DatabasePreferences The object or objects this property applies to.

BorderLineWidth

Double; read-write
The border line width.
BorderPath Property

Specifies the directory in which the border files are located.

See Also | Example

**Signature**

`object.BorderPath`

`object`

`PreferencesFiles` The object or objects this property applies to.

`BorderPath`

String; read-write
The drive and path for the drawing setup border .dwg files.

**Remarks**

The project path is C:\My DataPath\borders. By default, the data directory is C:\Documents and Settings\All Users\Application Data\Autodesk\AutoCAD Land Desktop <version number>\<release number>\Data. You must exit AutoCAD Land Desktop and restart for this change to take effect.
**BorderRightMargin Property**

Specifies the right margin for the border.

**Signature**

object(BorderRightMargin

object

  DatabasePreferences The object or objects this property applies to.

BorderRightMargin

  Double; read-write
  The border right margin.
BorderStyle Property

Specifies the border style to use for the drawing.

See Also | Example

Signature

object.BorderStyle

object

(DatabasePreferences) The object or objects this property applies to.

BorderStyle

eAeccBorderStyle enum; read-write
The type of the Border.

kBorderStyleLine: Insert a polyline

kBorderStyleUnscaledBlock: Insert a block at 1:1

kBorderStyleScaledBlock: Insert a block using DatabaseScale

kBorderStyleNone: No border
BorderTopMargin Property

Specifies the top margin for the border.

Signature

object.BorderTopMargin

object

DatabasePreferences The object or objects this property applies to.

BorderTopMargin

Double; read-write
The border top margin.
**Boundaries Property**

Returns the Boundaries collection.

**Signature**

object.Boundaries

object

`SurfaceInputs` The object or objects this property applies to.

Boundaries

`Boundaries` collection; read-only
The `Boundaries` collection.
Boundary Property

Returns the boundary of the Watershed.

See Also | Example

Signature

object.Boundary

object

WaterShed The object or objects this property applies to.

Boundary

Variant (array of doubles); read-only
An array of doubles representing the vertices of the boundary for the WaterShed.
BreakLines Property

Returns the BreakLines collection.

**Signature**

object.BreakLines

object

  **SurfaceInputs** The object or objects this property applies to.

BreakLines

  **BreakLines** collection; read-only
  The **BreakLines** collection.
CCWFlag Property

The counter-clockwise flag for the curve or parcel.

See Also | Example

Signature: Overview

1 AlignCurve

1 ParcelCurve

The counter-clockwise flag for the curve.

Signature

object.CCWFlag

object

AlignCurve The object or objects this property applies to.

CCWFlag

Boolean; read-only
TRUE: The curve is counter-clockwise.
FALSE: The curve is clockwise.

The counter-clockwise flag for the parcel.

Signature
object.CCWFlag

object

ParcelCurve
The object or objects this property applies to.

CCWFlag

Boolean; read-only
TRUE: The curve is counter-clockwise.
FALSE: The curve is clockwise.
CenterEasting Property

The Easting coordinate for the alignment curves or parcel curves centroid.

**Signature:** Overview

- **AlignCurve**
- **ParcelCurve**

The Easting coordinate for the curve's centroid.

**Signature**

```object.CenterEasting```

- **object**

  **AlignCurve** The object or objects this property applies to.

  **CenterEasting**
  
  Double; read-only
  
  The east coordinate for the center of the curve.

The Easting coordinate for the parcel's centroid.

**Signature**
object.CenterEastings

object

ParcelCurve
The object or objects this property applies to.

CenterEastings

Double; read-only
The east coordinate for the center of the curve.
CenterlineOffset Property (Civil Engineering Feature)

Returns the centerline offset of the cross section.

See Also | Example

Signature

object.CenterlineOffset

object

CrossSectionBlock The object or objects this property applies to.

CenterlineOffset

Double; read-only
The centerline offset of the cross section.
**CenterNorthing Property**

The Northing coordinate for the alignment curves or parcel curves center.

See Also | Example

**Signature: Overview**

* AlignCurve
* ParcelCurve

The Northing coordinate for the curve's center.

**Signature**

```csharp
object.CenterNorthing
```

- **object**
  - **AlignCurve** The object or objects this property applies to.

- **CenterNorthing**
  - Double; read-only
  - The north coordinate for the center of the curve.

The Northing coordinate for the parcel's center.

**Signature**

```csharp
object.CenterNorthing
```
object

ParcelCurve
The object or objects this property applies to.

CenterNorthing

Double; read-only
The north coordinate for the center of the curve.
CentroidEasting Property

Returns the centroid easting of the Parcel.

**Signature**

object.CentroidEasting

object

Parcel The object or objects this property applies to.

CentroidEasting

Double; read-only
The parcel's centroid easting value.
CentroidNorthing Property

The northing coordinate of the parcel centroid.

Signature

object.CentroidNorthing

  object
  Parcel The object or objects this property applies to.

  CentroidNorthing
  Double; read-only
  The parcel's centroid northing value.

See Also | Example
ChordDirection Property

The direction of the chord for the curve.

See Also | Example

**Signature**

object.ChordDirection

  object

  AlignCurve The object or objects this property applies to.

  ChordDirection

  Double; read-only
  The angle of the chord for the curve. The angle is in radians and measured counter-clockwise from the X axis (East).
**ChordLength Property**

The chord length of the alignment or parcel curve.

**Signature: Overview**

| AlignCurve |
| ParcelCurve |

The chord length of the alignment curve.

**Signature**

object.ChordLength

  object

      AlignCurve The object or objects this property applies to.

  ChordLength

      Double; read-only
      The chord length of the curve.

The chord length of the parcel curve.

**Signature**

object.ChordLength
object

ParcelCurve
The object or objects this property applies to.

ChordLength

Double; read-only
The chord length of the curve.
CivilDataFilesPath Property

Specifies the directory in which the civil engineering feature data is located.

See Also | Example

**Signature**

object.CivilDataFilesPath

object

PreferencesFiles The object or objects this property applies to.

CivilDataFilesPath

String; read-write
The drive and path for the civil engineering feature data files.

**Remarks**

By default, this path is C:\Documents and Settings\All Users\Application Data\Autodesk\AutoCAD Land Desktop <version number>\<release number>\Data, the same as the default path for the AutoCAD Land Desktop data path. You must exit AutoCAD Land Desktop and restart for this change to take effect.
**Code Property**

Returns the cross section point code or description key code.

See Also | Example

**Signature: Overview**

I [CrossSectionPointCode](#) (Civil Engineering Feature)

I [DescriptionKey](#)

---

Returns the cross section point code.

**Signature**

object.Code

    object

    [CrossSectionPointCode](#) The object or objects this property applies to.

    **Code**

    Integer; read-only

Returns the cross section point code.

---

Specifies the description key syntax.

**Signature**

object.Code
object

**DescriptionKey**
The object or objects this property applies to.

**Code**

String; read-only.
Returns the code for a description key, not including the path.
**Cogo Property**

The collection of cogo preferences for the project.

**See Also** | **Example**

**Signature**

object.Cogo

object

[PreferencesProject](#) The object or objects this property applies to.

Cogo

[PreferencesCogo](#); read-only
The [PreferencesCogo](#) object.
CogoPoints Property

Gets the CogoPoints collection.

See Also | Example

**Signature**

object.CogoPoints

object

Project The object or objects this property applies to.

CogoPoints

CogoPoints collection; read-only
The CogoPoints collection.

**Remarks**

The CogoPoints collection represents all of the CogoPoint objects in this project. This property is valid only for the ActiveProject.
ContourItems Property

Returns the ContourItems collection.

Signature

object.ContourItems

object

SurfaceInputs The object or objects this property applies to.

ContourItems

ContourItems collection; read-only
The ContourItems collection.
ContourStyle Property

Returns and set the Contour Style object for this contour.

See Also | Example

Signature

object.ContourStyle

object

AeccContour The object or objects this property applies to.

ContourStyle

AeccContourStyle object; read-write
A contour style from the dictionary.

Remarks

You can use this property to get the contour style currently applied to this contour, or apply a different style.

The Contour Style object represents an entry in the dictionary named "AECC_CONTOUR_STYLES". The Contour Style object exposes a Name property, which is the keyword. You can iterate through the dictionary as a collection to determine the contour styles available in the drawing.
ContourStyleName Property

Returns the Contour Style

See Also | Example

**Signature**

object.ContourStyleName

object

AeccContour The object or objects this property applies to.

ContourStyleName

String, read-write
Returns the Contour Style name.
ContourStylesPath Property

Specifies the directory in which the contour styles are located.

See Also | Example

Signature

object.ContourStylesPath

object

PreferencesFiles The object or objects this property applies to.

ContourStylesPath

String; read-write
The drive and path for the contour style files.

Remarks

By default, the project path is C:\Documents and Settings\All Users\Application Data\Autodesk\AutoCAD Land Desktop <version number>\<release number>\Data\Contours. You must exit AutoCAD Land Desktop and restart for this change to take effect.
## CoordinatePrecision Property

Specifies the coordinate precision for the drawing.

**Signature**

```
object.CoordinatePrecision
```

**object**

The object or objects this property applies to.

**CoordinatePrecision**

*Long; read-write*

The number of decimals to the right of the decimal point.

**Remarks**

This is used to display all Northing/Easting coordinate information.

The precision settings are used only for labeling and listing values, not actual computations. AutoCAD Land Desktop commands always calculate all numbers up to the highest internal precision.
The coordinate properties for various objects.

**Signature: Overview**

- **AeccContour**
- **Boundary**
- **BreakLine**
- **CogoPoint**
- **ContourItem**
- **CrossSectionBlock** (Civil Engineering Feature)
- **Edge**
- **ElevationContour**
- **Face**
- **ProfileBlock** (Civil Engineering Feature)

Returns the vertices of the Contour in XYZ as doubles.

**Signature**
object.Coordinates

object

AeccContour The object or objects this property applies to.

Coordinates

Variant (three-element array of doubles); read-write
The array of points.

Returns the vertices of the Boundary in ENZ as doubles.

Signature

object.Coordinates

object

Boundary
The object or objects this property applies to.

Coordinates

Variant (three-element array of doubles); read-only
The array of points.

Returns the vertices of the BreakLine in ENZ as doubles

Signature

object.Coordinates

object

BreakLine
The object or objects this property applies to.

Coordinates

Variant (three-element array of doubles); read-only
The array of points.
Remarks
To modify the points for a BreakLine, you must deleted the BreakLine and add a new one.

Returns the point coordinates in XYZ as doubles.

Signature
object.Coordinates

object

CogoPoint
The object or objects this property applies to.

Coordinates
Variant: (3 element array of doubles); read-only
The point coordinate

Returns the vertices of the ContourItem in ENZ as doubles.

Signature
object.Coordinates

object

ContourItem
The object or objects this property applies to.

Coordinates
Variant (three-element array of doubles); read-only
The vertices of the contour.

Returns the alignment cross section coordinates.

Signature
object.Coordinates

object

**CrossSectionBlock**
The object or objects this property applies to.

**Coordinates**

Variant (two-element array of doubles); read-only
The array of X and Y values.

**Remarks**
The coordinates represent the AutoCAD X and Y position of the alignment cross section where the user selected the bottom insertion point.

Returns the vertices of the Edge in ENZ as doubles.

**Signature**

object.Coordinates

object

**Edge**
The object or objects this property applies to.

**Coordinates**

Variant (6 element array of doubles); read-only
The point coordinate for the start and end of the edge

Returns the vertices of the ElevationContour in ENZ as doubles.

**Signature**

object.Coordinates Points

object
**ElevationContour**
The object or objects this property applies to.

**Coordinates**
Variant (array of doubles); read-only
Contains an array of doubles in the format of Easting, Northing, Elevation, etc.

Returns the vertices of the face in ENZ as doubles.

**Signature**

object.Coordinates

object

**Face**
The object or objects this property applies to.

**Coordinates**
Variant (9 element array of doubles); read-only
The point coordinates for the three points of the face triangle.

**Remarks**
Returns the coordinates for the face. The vertices are numbered 0, 1, and 2. They are returned in counter-clockwise order.

Returns the Coordinates of the Alignment Profile as AutoCAD X and Y values.

**Signature**

object.Coordinates

object

**ProfileBlock**
The object or objects this property applies to.

Coordinates

Variant (two-element array of doubles); read-only
The array of X and Y values.

Remarks

The coordinates represent the AutoCAD X and Y position of the lower left corner of the alignment profile where the user selected the starting point.
**CoordinateZone Property**

Specifies the coordinate zone.

**Signature: Overview**

<table>
<thead>
<tr>
<th>DatabasePreferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEMFile</td>
</tr>
</tbody>
</table>

Specifies the preferences coordinate zone.

**Signature**

object.CoordinateZone

object

    DatabasePreferences  The object or objects this property applies to.

    CoordinateZone

        String; read-write
        The coordinate system syntax for the zone.

**Remarks**

Drawings in a project can have the same or different coordinate zones assigned to them. To work in real-world coordinates, you must establish a current zone for the drawing.
The coordinate zone code for the DEM file.

**Signature**

`object.CoordinateZone`

object

`DEMFile`

The object or objects this property applies to.

CoordinateZone

String; read-only

The coordinate zone code for the DEM file.

**Remarks**

The CoordinateZone property is optional, and can only be set with the `Add` method of the `DEMFiles` collection.
Count Property

Gets the number of items in the collection.

Signature

object.Count

object

All Collections The object or objects this property applies to.

Count

Long; read-only
The number of items in the collection.
Course Property

Returns the Course of the parcel entity.

See Also | Example

Signature

object.Course

object

ParcelEntity, ParcelCurve, ParcelLine

The object or objects this property applies to.

Course

Double; read-only

The course In for the parcel. The course angle is in radians and measured counter-clockwise from the X axis (East).
CourseIn Property

Returns the Course In of the Parcel.

See Also | Example

**Signature**

object.CourseIn

  object
  **ParcelCurve** The object or objects this property applies to.

  CourseIn

  Double; read-only
  The course In for the parcel. The course angle is in radians and measured counter-clockwise from the X axis (East).
CourseOut Property

Returns the Course Out of the Parcel.

See Also | Example

Signature

object.CourseOut

object

ParcelCurve The object or objects this property applies to.

CourseOut

Double; read-only
The course In for the parcel. The course angle is in radians and measured counter-clockwise from the X axis (East).
CrossSection Property (Civil Engineering Feature)

The collection of cross section preferences for the project.

See Also | Example

Signature

object.CrossSection

object

PreferencesProject The object or objects this property applies to.

CrossSection

PreferencesCrossSection; read-only

The PreferencesCrossSection object.
CrossSectionBlocks Property

Gets the CrossSectionBlocks object.

**Signature**

object.CrossSectionBlocks

- **object**
  - Document The object or objects this property applies to.

- CrossSectionBlocks
  - CrossSectionBlocks object; read-only
  - The CrossSectionBlocks object.
CrossSectionPointCodes Property (Civil Engineering Feature)

Returns the cross sections point codes collection

See Also | Example

Signature

object.CrossSectionPointCodes

object

CrossSection

CrossSectionPointCodes

The object or objects this property applies to.

CrossSectionPointCodes

CrossSectionPointCodes collection; read-only

The CrossSectionPointCodes collection.
CrossSections Property (Civil Engineering Feature)

Gets the alignment cross sections collection.

**Signature**

object.CrossSections

object

Alignment The object or objects this property applies to.

CrossSections

CrossSections collection; read-only
The CrossSections collection.
CrossSectionSurfaces Property (Civil Engineering Feature)

Returns the cross sections surfaces collection

Signature

object.CrossSectionSurfaces

object

CrossSection The object or objects this property applies to.

CrossSectionSurfaces Collection

CrossSectionSurfaces collection; read-only
The CrossSectionSurfaces collection.
CurrentAlignment Property

Sets and gets the current alignment.

See Also | Example

Signature

object.CurrentAlignment

object

Alignments The object or objects this property applies to.

CurrentAlignment

String; read-write
The current alignment name.

Remarks

All alignments initially are loaded with read only permission. Only the current alignment can have write permissions. However, another user could have already locked that alignment. If this happens, read permissions will be granted for the current alignment.

By changing the current alignment, you are a) locking / unlocking alignments and b) setting the current alignment in the AutoCAD Land Desktop menus.

Setting CurrentAlignment to an empty string removes the current alignment status.
CurrentSurface Property

Gets / sets the current Surface.

See Also | Example

Signature

object.CurrentSurface

object

Surfaces The object or objects this property applies to.

CurrentSurface

String; read-write
The current surface set in project.

Remarks

Only one surface can be considered current in project. This surface is both opened and locked.
CurveCode Property (Civil Engineering Feature)

Returns the superelevation curve code.

See Also | Example

Signature

object.CurveCode

object

**Superelevation** The object or objects this property applies to.

CurveCode

eAeccSectionSECurveCode; read-only

kSERightHandCurve:  Right hand curve

kSLELeftHandCurve:  Left hand curve
CurveLabelIncrement Property (Civil Engineering Feature)

Returns the distance between vertical curve elevation labels.

See Also | Example

Signature

object.CurveLabelIncrement

object

ProfileBlock The object or objects this property applies to.

CurveLabelIncrement

Double; read-only
The Curve Label increment for the Profile.
CurveLength Property (Civil Engineering Feature)

Returns the Curve Length of the PVI.

See Also | Example

**Signature**

object.CurveLength

object

**PVI** The object or objects this property applies to.

CurveLength

Double; read-only
The Curve Length for the PVI.
### DatabaseScale Property

Specifies the database scale at which the horizontal axis of the drawing is displayed.

**Signature**

object.DatabaseScale

- **object**
  - DatabasePreferences The object or objects this property applies to.
- **DatabaseScale**
  - Double; read-write
  - The database (horizontal) scale. For example, for a 1:200 scale, type 200.

**Remarks**

The database (horizontal) scale controls the size of annotation placed in the drawing, including text, scaled blocks and special lines. It does not affect the line lengths or point coordinates because they are always defined in real world coordinates, not to any scale. Neither does it affect any design data in the drawing or project files.

If you change the DatabaseScale property in the middle of a drawing session, then any annotation added subsequently is scaled accordingly.
**DataPath Property**

Specifies the directory in which the AutoCAD Land Desktop data files are installed.

---

**Signature**

object.DataPath

object

[PreferencesFiles](#) The object or objects this property applies to.

DataPath

String; read-only

The drive and path for the data files.

---

**Remarks**

This value is read-only and cannot be changed.
DatumElevation Property (Civil Engineering Feature)

Returns the datum elevation for the alignment profile or cross section.

See Also | Example

Signature: Overview

1 CrossSectionBlock (Civil Engineering Feature)

1 ProfileBlock (Civil Engineering Feature)

Returns the datum elevation for the alignment cross section.

Signature

object.DatumElevation

object

CrossSectionBlock The object or objects this property applies to.

DatumElevation

Double; read-only

The datum elevation for the alignment cross section.

Returns the datum elevation for the alignment profile.

Signature

object.DatumElevation
object

ProfileBlock
The object or objects this property applies to.

DatumElevation

Double; read-only
The datum elevation for the alignment profile.
Delta Property

The delta value for a AlignCurve, AlignSpiral, or ParcelCurve.

See Also | Example

**Signature: Overview**

- AlignCurve, AlignSpiral
- ParcelCurve

The delta value for the alignment curve or spiral.

**Signature**

object.Delta

object

  AlignCurve, AlignSpiral
  The object or objects this property applies to.

Delta

  Double; read-only
  The internal (included) angle for the curve in radians.

The delta value for the parcel curve.

**Signature**
object.Delta

object

ParcelCurve
The object or objects this property applies to.

Delta

Double; read-only
The internal (included) angle for the curve in radians.
**DEMFiles Property**

Returns the DEMFiles collection.

**Signature**

object.DEMFiles

object

[SurfaceInputs](#) The object or objects this property applies to.

DEMFiles

[DEMFiles](#) collection; read-only

The [DEMFiles](#) collection.
Description Property

The description properties for various objects.

**Signature: Overview**

- **AeccPoint**
- **Alignment**
- **Boundary**
- **BreakLine**
- **CrossSectionPointCode** (Civil Engineering Feature)
- **PointGroup**
- **Project**
- **Prototype**
- **Surface**

The description for the point object.

**Signature**

```object.Description```
object

AeccPoint The object or objects this property applies to.

Description

String; read-write
The description for the point object.

The alignment description.

Signature

object.Description

object

Alignment
The object or objects this property applies to.

Description

String; read-write
The description for the alignment.

The description for the Boundary.

Signature

object.Description

object

Boundary
The object or objects this property applies to.

Description

String; read-write
The description of the boundary.
The description for the BreakLine

**Signature**

object.Description

    object

    BreakLine
    The object or objects this property applies to.

    Description

        String; read-write
        The description for the BreakLine.

The description for the cross sections point code.

**Signature**

object.Description

    object

    CrossSectionPointCode
    The object or objects this property applies to.

    Description

        String; read-only
        The description for the point code.

Specifies the description override for the project database point group.

**Signature**

object.Description

    object

    PointGroup
    The object or objects this property applies to.
Description

String; read-write

Remarks

If the Description property is set, then the DescriptionXDRef property is cleared.

Specifies the prototype used for the project.

Signature

object.Description

object

Prototype

String; read-write
The prototype used for this project.

The prototype description.

Signature

object.Description

object

Prototype

The object or objects this property applies to.

Description

String; read-write
The prototype description.
Remarks

Use this property to change the description of a prototype.

The description for the Surface.

Signature

object.Description

    object

        Surface

The object or objects this property applies to.

Description

    String; read-write

The description for the surface.

Remarks

The Surface description can be up to 255 characters.
DescriptionFormat Property

Specifies the description key format.

See Also | Example

**Signature**

object.DescriptionFormat

object

DescriptionKey The object or objects this property applies to.

DescriptionFormat

String; read-write

Returns and sets the description format.
DescriptionKeyFiles Property

Gets the DescriptionKeyFiles collection.

Signature

object.DescriptionKeyFiles

object

Project The object or objects this property applies to.

DescriptionKeyFiles

DescriptionKeyFiles collection; read-only
The DescriptionKeyFiles collection.

Remarks

The DescriptionKeyFiles collection represents all of the DescriptionKeyFiles objects in this project. This property is valid only for the ActiveProject.
DescriptionLayer Property

Specifies the description key layer.

**Signature**

object.DescriptionLayer

  object

  **DescriptionKey** The object or objects this property applies to.

DescriptionLayer

String; read-write

Returns and sets the description layer.
**DescriptionOverride Property**

Specifies if the description for the project database point will be overridden from the Point Group.

**See Also | Example**

**Signature**

object.DescriptionOverride

object

**PointGroup** The object or objects this property applies to.

**DescriptionOverride**

Boolean; read-write

TRUE: The Cogo project point description will be set by the Point Group.

FALSE: The Cogo project point description will not be set by the Point Group.

**Remarks**

When you set this value to FALSE, the Description property will be cleared.
DescriptionXDRef Property

Specifies the description override for the project database point group.

See Also | Example

Signature

object.DescriptionXDRef

object

PointGroup The object or objects this property applies to.

DescriptionXDRef

String; read-write
The name of the external database reference.

Remarks

The description will be taken from the external database reference (XDRef). If the DescriptionXDRef property is set, then the Description property is cleared.
**Direction Property**

The Direction of the ProfileBlock or alignment tangent.

See Also | Example

**Signature: Overview**

1 AlignTangent

1 ProfileBlock (Civil Engineering Feature)

---

The direction of the alignment tangent.

**Signature**

object.Direction

object

   *AlignTangent* The object or objects this property applies to.

**Direction**

Double; read-only
The polar direction of the tangent.

---

The direction of the alignment profile.

**Signature**

object.Direction
object

(ProfileBlock)
The object or objects this property applies to.

Direction

String; read-only
The direction of the alignment profile.
Documents Property

Gets the Document collection.

See Also | Example

Signature

object/Documents

object

AeccApplication The object or objects this property applies to.

Documents

Documents collection; read-only

Remarks

The Documents collection allows you to access all the documents, or drawings, in the current AutoCAD Land Desktop session.
DrainsInto Property

Returns an array of Ids that the Watershed drains into.

See Also | Example

Signature

object.DrainsInto

object

WaterShed The object or objects this property applies to.

DrainsInto

Variant (array of longs); read-only
An array of WaterShed Ids that the current WaterShed drains into.
**DrawingPath Property**

Specifies the drawing path of the project.

**Signature**

```
object.DrawingPath
```

object

*Project* The object or objects this property applies to.

DrawingPath

String; read-write
The drawing path used by the project.

**Remarks**

The initial value of this property is empty - the default drawing path to save drawings in is the `\dwg` subfolder.
Drawings Property

Gets the Drawings collection.

**Signature**

```plaintext
object.Drawings

object

Project The object or objects this property applies to.

Drawings

Drawings collection; read-only
The Drawings collection.

**Remarks**

The Drawings collection represents all drawings in the DrawingPath for this project.
**DrawingSetupPath Property**

Specifies the directory in which the drawing setup files are located.

**See Also** | **Example**

**Signature**

object.DrawingSetupPath

object

PreferencesFiles The object or objects this property applies to.

DrawingSetupPath

String; read-write

The drive and path for the drawing setup files.

**Remarks**

By default, this path is C:\Documents and Settings\All Users\Application Data\Autodesk\AutoCAD Land Desktop <version number>\<release number>\Data\setup. You must exit AutoCAD Land Desktop and restart for this change to take effect.
Easting Property

The Easting coordinates properties for various objects.

See Also | Example

Signature: Overview

| AeccPoint
| CogoPoint
| TinPoint

The Easting coordinate of the point.

Signature

object.Easting

  object
  AeccPoint The object or objects this property applies to.

Easting

  Double; read-write
  The Easting coordinate of the point.

Remarks

The Easting coordinate is derived from the WCS X coordinate with the BasePoint and NorthRotation applied.
The east coordinate for the point.

**Signature**

object.Easting

object

*CogoPoint*

The object or objects this property applies to.

Easting

Double; read-write

---

The Easting coordinate for the TinPoint.

**Signature**

object.Easting

object

*TinPoint*

The object or objects this property applies to.

Easting

Double; read-only

The Easting coordinate.
Edges Property

Returns the Edges collection.

See Also | Example

Signature

object.Edges

object

SurfaceOutputs The object or objects this property applies to.

Edges

Edges collection; read-only
The Edges collection.
EGPrecision Property (Civil Engineering Feature)

Returns the existing ground precision for the alignment profile or cross section.

See Also | Example

**Signature: Overview**

- **CrossSectionBlock** (Civil Engineering Feature)
- **ProfileBlock** (Civil Engineering Feature)

Returns the existing ground precision for the alignment cross section.

**Signature**

object.EGPrecision

object  

**CrossSectionBlock** The object or objects this property applies to.

**EGPrecision**

Integer; read-only

The existing ground precision for the alignment cross section.

Returns the existing ground precision for the alignment profile.
**Signature**

object.EGPrecision

- **object**
  - **ProfileBlock**
    - The object or objects this property applies to.

- **EGPrecision**
  - Integer; read-only
    - The existing ground precision for the alignment profile.
EGProfiles Property (Civil Engineering Feature)

Gets the existing ground profiles collection.

See Also | Example

**Signature**

object.EGProfiles

object

Alignment The object or objects this property applies to.

EGProfiles

EGProfiles collection; read-only
The **EGProfiles** collection.
Elevation Property

The elevation property for various objects.

See Also | Example

Signature: Overview

- AeccContour
- AeccPoint
- CogoPoint
- CrossSectionPointCode (Civil Engineering Feature)
- ElevationContours
- PointGroup
- PVI (Civil Engineering Feature)
- TinPoint

The elevation of the AeccContour.

Signature

object.Elevation

  object

  AeccContour The object or objects this property
The elevation of the AnccPoint.

**Signature**

object.Elevation

object

**AnccPoint**
The object or objects this property applies to.

Elevation

Double; read-write
The elevation of the point.

**Remarks**
The Elevation property is always the same as the WCS Z coordinate.

The elevation for the CogoPoints.

**Signature**

object.Elevation

object

**CogoPoint**
The object or objects this property applies to.

Elevation

Double; read-write

**Remarks**
The Elevation property is the raw elevation. PointGroup Overrides will not be applied to this property.

The elevation for the cross section point code.

**Signature**

object.Elevation

object

CrossSectionPointCode

The object or objects this property applies to.

Elevation

Double; read-only

The elevation for the cross section point code.

The elevation for the ElevationContours collection.

**Signature**

object.Elevation

object

ElevationContours

The object or objects this property applies to.

Elevation

Double; read-write

The elevation of the Contours collection.

**Remarks**

When the elevation is set, the ElevationContours collection will be regenerated with the contours for that elevation. All contours in the collection are of the same elevation.
Specifies the elevation override for the project database point group.

**Signature**

object.Elevation

object

**PointGroup**
The object or objects this property applies to.

Elevation

String; read-write

**Remarks**

If the Elevation property is set, then the **ElevationXDRef** property is cleared.

Returns the Elevation of the PVI.

**Signiture**

Object.Elevation

object

**PVI**
The object or objects this property applies to.

Elevation

Double; read-only

The Elevation for the PVI.

The Elevation for the TinPoint

**Signature**

object.Elevation
object

*TinPoint*

The object or objects this property applies to.

Elevation

Double; read-only

The elevation for the point.
ElevationContours Property

Returns the GenerateContours collection.

See Also | Example

Signature

object.ElevationContours

object

SurfaceOutputs The object or objects this property applies to.

ElevationContours

ElevationContours collection; read-only

The ElevationContours collection.
ElevationOverride Property

Specifies if the elevation for the project database point will be overridden from the Point Group.

See Also | Example

Signature

object.ElevationOverride

object

PointGroup The object or objects this property applies to.

ElevationOverride

Boolean; read-write

TRUE: The Cogo project point elevation will be set by the Point Group.

FALSE: The Cogo project point elevation will not be set by the Point Group.

Remarks

When you set this value to FALSE, the Elevation property will be cleared.
ElevationPrecision Property

Specifies the elevation precision for the drawing.

See Also | Example

Signature

object.ElevationPrecision

object

DatabasePreferences The object or objects this property applies to.

ElevationPrecision

Long; read-write
The number of decimals to the right of the decimal point.

Remarks

The precision settings are used only for labeling and listing values, not actual computations. AutoCAD Land Desktop commands always calculate all numbers up to the highest internal precision.
**ElevationXDRef Property**

Specifies the elevation override for the project database point group.

See Also | Example

**Signature**

object.ElevationXDRef

object

PointGroup The object or objects this property applies to.

ElevationXDRef

String; read-write

The name of the external database reference.

**Remarks**

The elevation will be taken from the external database reference (XDRef). If the ElevationXDRef property is set, then the Elevation property is cleared.
EndDirection Property

The ending direction of the curve.

See Also | Example

Signature

object.EndDirection

object

AlignCurve The object or objects this property applies to.

EndDirection

Double; read-only
The end angle of the curve.

Remarks

The direction is in radians and measured counter-clockwise from the X axis (East).
EndEasting Property

The Easting coordinate for the end of the individual object entities.

See Also | Example

Signature: Overview

| AlignEntity, AlignCurve, AlignSpiral, AlignTangent |
| ParcelEntity, ParcelCurve, ParcelLine |

The Easting coordinate for the end of the individual alignment entities.

Signature

object.EndEasting

object

AlignEntity, AlignCurve, AlignSpiral, AlignTangent
The object or objects this property applies to.

EndEasting

Double; read-only
The Easting coordinate for the end point.

The Easting coordinate for the end of the individual parcel entities.

Signature

object.EndEasting
object

ParcelEntity, ParcelCurve, ParcelLine
The object or objects this property applies to.

EndEasting

Double; read-only
The Easting coordinate for the end point.
EndingStation Property

The object's ending station.

See Also | Example

Signature: Overview

- AlignEntity, AlignCurve, AlignSpiral, AlignTangent
- Alignment
- ProfileBlock (Civil Engineering Feature)

The EndingStation for individual entities in an alignment.

Signature

object.EndingStation

object

AlignEntity, AlignCurve, AlignSpiral, AlignTangent

The object or objects this property applies to.

StartingStation

Double; read-only

The ending station for the object.

The EndingStation for the overall alignment.

Signature
object.EndingStation

object

**Alignment**
The object or objects this property applies to.

EndingStation

Double; read-only
The ending station for the object.

Returns the EndingStation for the Profile.

**Signature**

object.EndStation

object

**ProfileBlock**
The object or objects this property applies to.

EndStation

Double; read-only The Ending Station for the Profile.
EndNorthing Property

The Northing coordinate for the end of the individual object entity.

See Also | Example

Signature: Overview

| AlignEntity, AlignCurve, AlignSpiral, AlignTangent |
| ParcelEntity, ParcelCurve, ParcelLine |

The Northing coordinate for the end of the individual alignment entities.

Signature

object.Insert

object

AlignEntity, AlignCurve, AlignSpiral, AlignTangent
The object or objects this property applies to.

EndNorthing

Double; read-only
The Northing coordinate for the end point.

The Northing coordinate for the end of the individual parcel entities.

Signature

object.Insert
object

*ParcelEntity*, *ParcelCurve*, *ParcelLine*

The object or objects this property applies to.

EndNorthiing

Double; read-only
The Northing coordinate for the end point.
ExceedBoth Property

If true, then both minimum depression settings must be met.

See Also | Example

Signature

object.ExceedBoth

object

WaterSheds The object or objects this property applies to.

ExceedBoth

Boolean; read-only
TRUE: Both minimum depression conditions must be met.
FALSE: Both minimum depression conditions do not need to be met.
**ExtEasting Property**

The Easting coordinate for the External point (Ext).

See Also | Example

**Signature**

object.ExtEasting

object

*AlignSpiral* The object or objects this property applies to.

ExtEasting

Double; read-only
The Easting coordinate for the External point.

**Remarks**

The External point is used for compound spirals, where the TS point is not on the spiral. This would mark the point where the spiral begins or ends.
ExternalSecant Property

The external secant length for the curve.

See Also | Example

Signature

object.ExternalSecant

object

AlignCurve The object or objects this property applies to.

ExternalSecant

Double; read-only
The external secant length for the curve.
ExtNorthing Property

The Northing coordinate for the External point (Ext).

Signature

object.ExtNorthing

object  

AlignSpiral The object or objects this property applies to.

ExtNorthing

Double; read-only
The Northing coordinate for the External point.

Remarks

The External point is used for compound spirals, where the TS point is not on the spiral. This would mark the point where the spiral begins or ends.
Faces Property

Returns the Faces collection.

See Also | Example

Signature

object.Faces

object

SurfaceOutputs The object or objects this property applies to.

Faces

Faces collection; read-only
The Faces collection.
FacetDeviation Property

Gets and sets the facet deviation.

See Also | Example

Signature

object.FacetDeviation

object

DatabasePreferences The object or objects this property applies to.

FacetDeviation

Double; read-write
The facet deviation.
FGPrecision Property (Civil Engineering Feature)

Returns the finished ground precision for the alignment profile or cross section.

See Also | Example

Signature: Overview

- CrossSectionBlock (Civil Engineering Feature)
- ProfileBlock (Civil Engineering Feature)

Returns the finished ground precision for the alignment cross section.

Signature

object.FGPrecision

object

CrossSectionBlock The object or objects this property applies to.

FGPrecision

Integer; read-only
The finished ground precision for the alignment profile.

Returns the finished ground precision for the alignment profile.
**Signature**

object.FGPrecision

profileBlock
The object or objects this property applies to.

FGPrecision

Integer; read-only
The finished ground precision for the alignment profile.
FGProfiles Property (Civil Engineering Feature)

The proposed vertical alignment's entities collection.

**Signature**

```
object.ProposedVerticalAligns

object
```

*Alignment* The object or objects this property applies to.

```
FGProfiles

FGProfiles collection; read-only

The FGProfiles collection.
```
File Property

The filename of the file lock.

See Also | Example

Signature

object.File

object

FileLock The object or objects this property applies to.

File

String; read-only
The filename and path of the lock file.
FileLocks Property

Gets the FileLocks collection.

Signature

object.FileLocks

object

Project The object or objects this property applies to.

FileLocks

FileLocks collection; read-only
The FileLocks collection.

Remarks

The FileLocks collection represents all file locks for this project.
Files Property

Gets the PreferencesFiles object.

See Also | Example

Signature

object.Files

object

Preferences The object or objects this property applies to.

Files

PreferencesFiles object; read-only
Gets the PreferencesFiles object.

Remarks

This object specifies the paths used by the program. Some paths are established during installation and are exposed as read-only properties.
FirstTimeDrawingSetup Property

Selects how new drawings are setup.

See Also | Example

**Signature**

object.FirstTimeDrawingSetup

  object

  PreferencesUser The object or objects this property applies to.

FirstTimeDrawingSetup
eAeccFirstTimeDrawingSetup enum; read-write
The type of drawing setup.

  kDrawingSetupWizard: Use the Drawing Setup Wizard

  kDrawingSetupCommand: Use the Drawing Setup command

  kAutoloadSetupFile: Automatically load a setup file

**Remarks**

The initial value for this property is kDrawingSetupWizard.
If you choose kAutoloadSetupFile, the file used is specified by the FirstTimeDrawingSetupFile property.
FirstTimeDrawingSetupFile Property

The drawing setup file loaded automatically for a new drawing.

See Also | Example

Signature

object.FirstTimeDrawingSetupFile

object

PreferencesUser The object or objects this property applies to.

FirstTimeDrawingSetupFile

String; read-write
The name of the setup file that will be loaded.
Format Property

The format for the PointFile.

See Also | Example

Signature

object.Format

object

PointFile The object or objects this property applies to.

Format

String; read-only
The format for the input file.

Remarks

The input file must conform to the standard NEZ (space-delimited) file format.
FormatsPath Property

Specifies the directory in which the point import and export formats are located.

See Also | Example

Signature

object.FormatPath

object

PreferencesFiles The object or objects this property applies to.

FormatPath

String; read-write
The drive and path for the point import and export format files.

Remarks

By default, the project path is C:\Documents and Settings\All Users\Application Data\Autodesk\AutoCAD Land Desktop <version number>\<release number>\Data\Format Manager. This path is not created during install. The commands that support point import and export formats creates this path when required. You must exit AutoCAD Land Desktop and restart for this change to take effect.
**FullDescription Property**

Returns the full description for the Cogo Point.

### Signature

object.FullDescription

object

[CogoPoint](#) The object or objects this property applies to.

**FullDescription**

String; read-only

### Remarks

The FullDescription includes DescriptionKey Overrides.
**FullName Property**

The full name includes the path and file name.

**Signature: Overview**

- DescriptionKeyFile
- Drawing
- Project

The full name for the DescriptionKeyFile for a project. The full name includes the path and file name.

**Signature**

```
object.FullName
```

```
object DescriptionKeyFile;
```

The object or objects this property applies to.

```
FullName
```

String; read-only

Gets the name of the drawing, including the path.

**Signature**
object.FullName


doct

Drawing
The object or objects this property applies to.

FullName

String; read-only
The name and path of the application or document.

Remarks

This value is read-only and cannot be changed.

Specifications the name and path of the project.

Signature

object.FullName

object

Project
The object or objects this property applies to.

FullName

String; read-only
The name and path of the project.

Remarks

This property returns the fully qualified path and name of the project.
GridEastings Property

The grid Easting for the COGO project database point.

See Also | Example

Signature

object/GridEastings

object CogoPoint The object or objects this property applies to.

GridEastings

Double; read-only
GridNorthing Property

The grid Northing for the COGO project database point.

Signature

object/GridNorthing

object

CogoPoint The object or objects this property applies to.

GridNorthing

Double; read-only
GroupName Property

Gets / sets the group name for various objects.

Signature: Overview

- **CogoPoint**
- **PointGroup**

Gets / sets the group name for the Point.

**Signature**

```plaintext
object.GroupName
```

- **object**: `CogoPoint` The object or objects this property applies to.

- **GroupName**: String; read-write
  Returns and sets the group name for the CogoPoints collection.

**Remarks**

If no GroupName has been set, then the `OverrideDescription` and `OverrideName` will be empty. `OverrideElevation` will be -1E20. If
GroupName has been set, then these values will reflect the PointGroup overrides. If the PointGroup does not have an override for a field, the field will not change.

Gets / sets the group name for the PointGroup.

**Signature**

`object.GroupName`

- `object`
  - **PointGroup**
    - The object or objects this property applies to.

- `GroupName`
  - String; read-write
    - The group name for the PointGroup.
Height Property (Civil Engineering Feature)

Returns the height of the alignment cross section.

**Signature**

object.Height

  object [CrossSectionBlock] The object or objects this property applies to.

  Height

    Double; read-only
    The height of the alignment cross section.
HelpPath Property

Specifies the directory in which the help files are installed.

See Also | Example

**Signature**

object.HelpPath

object

PreferencesFiles The object or objects this property applies to.

HelpPath

String; read-only
The drive and path for the help files.
Id Property

The identification number for the Boundary, BreakLine, ContourItem or Watershed.

Signature: Overview

- Boundary
- BreakLine
- ContourItem
- WaterShed

An identification number for the Boundary.

Signature

object.Id

object

**Boundary** The object or objects this property applies to.

Id

Long: read-only
The identification number of the boundary.

Remarks
The Boundaries class uses the Boundary Id to delete and find an instance of a boundary.

The identification number for the BreakLine.

**Signature**

`object.Id`

`object`  
`BreakLine`  
The object or objects this property applies to.

`Id`

`Long; read-only`  
The identifier for the BreakLine.

An identification number for the ContourItem.

**Signature**

`object.Id`

`object`  
`ContourItem`  
The object or objects this property applies to.

`Id`

`Long: read-only`  
The identification number of the contour.

**Remarks**

The ContourItems class uses the ContourItem Id to delete and find an instance of a contour.

An identification number for the WaterShed.
**Signature**

`object.Id`

`object`  

WaterShed
The object or objects this property applies to.

`Id`

Long: read-only  
The identification number of the WaterShed.
Inputs Property

Returns the SurfaceInputs object for the surface.

See Also | Example

Signature

object.Inputs

object

Surface The object or objects this property applies to.

Inputs

SurfaceInputs; read-only

All inputs for the surface.
IsBreakLine Property

Is the Boundary considered a BreakLine?

See Also | Example

Signature

object.IsBreakLine

object

Boundary The object or objects this property applies to.

IsBreakLine

Boolean; read-write
TRUE: The Boundary will be considered as a breakline
FALSE: The boundary will not be considered as a breakline.

Remarks

If the Boundary is considered a breakline, faces will be generated along the Boundary.
IsNameSupported Property

Returns whether point names are supported or not.

See Also | Example

**Signature**

object.IsNameSupported

object  

[CogoPoints](#) The object or objects this property applies to.

IsNameSupported  

Boolean; read-only

TRUE: Point names are supported.
FALSE: Point names are not supported.
**IsVisible Property**

Returns if the Face is visible.

---

**Signature**

object.IsVisible

- object
- **Face** The object or objects this property applies to.
- IsVisible

  Boolean; read-only
  TRUE: If the Face is visible.
  FALSE: If the Face is not visible.

---

**Remarks**

Portions of the Surface can be made invisible by using boundaries. This does not mean that the Face does not exist in the Surface. It only means that the face is not shown when importing the Surface.

---
K Property

The spiral's K value.

Signature

object.K

object

AlignSpiral

The object or objects this property applies to.

K

Double; read-only

The spiral's K value.
Keywords Property

Specifies the keywords for the project.

See Also | Example

Signature

object.Keywords

object

Project The object or objects this property applies to.

Keywords

String; read-write
The keywords for the project.
Label Property

The name of the locked datafile or folder.

Signature

object.Label

  object

    FileLock The object or objects this property applies to.

  Label

    String; read-only
    The label of the file lock.

See Also | Example
## LabelPoints Property

Returns the contour label points.

### See Also | Example

### Signature

```plaintext
object.LabelPoints
```

**object**

AeccContour The object or objects this property applies to.

**LabelPoints**

Variant (three-element array of doubles); read-only

The array of points for the contour labels.
LabelStyle Property

Specifies the label style override for the project database point group.

**Signature**

object.LabelStyle

object

PointGroup The object or objects this property applies to.

LabelStyle

String; read-write
The label style override for the project database point group.
**LabelStyleOverride Property**

Specifies if the label style for the project database point will be overridden from the Point Group.

**Signature**

```
object.LabelStyleOverride
```

object

**PointGroup** The object or objects this property applies to.

LabelStyleOverride

Boolean; read-write

TRUE: The point group label style will be overridden.
FALSE: The point group label style will not be overridden.
LabelStylePath Property

Specifies the directory in which the line, curve, spiral and point label style files are located.

See Also | Example

Signature

object.LabelStylePath

object

PreferencesFiles The object or objects this property applies to.

LabelStylePath

String; read-write
The drive and path for the label style files.

Remarks

By default, the project path is C:\Documents and Settings\All Users\Application Data\Autodesk\AutoCAD Land Desktop <version number>\<release number>\Data\labels. You must exit AutoCAD Land Desktop and restart for this change to take effect.
LabelStyleXDRef Property

Specifies the label style override for the project database point group.

See Also | Example

Signature

object.LabelStyleXDRef

object

(PointGroup) The object or objects this property applies to.

LabelStyleXDRef

String; read-write
The label style override for the project database point group.
LastUsedDwg Property

The last used drawing.

See Also | Example

Signature

object.LastUsedDwg

object

PreferencesUser The object or objects this property applies to.

LastUsedDwg

String; read-only
The name of the drawing that was last used.
LastUsedDwgPath Property

The last used drawing path.

See Also | Example

Signature

object.LastUsedDwgPath

object

PreferencesUser The object or objects this property applies to.

LastUsedDwgPath

String; read-only
The name of the drawing path that was last used.
LastUsedProj Property

The last used project.

See Also | Example

Signature

object.LastUsedProj

object

PreferencesUser The object or objects this property applies to.

LastUsedProj

String; read-only
The name of the project that was last used.
LastUsedProjPath Property

The last used project path.

See Also | Example

Signature

object.LastUsedProjPath

object

PreferencesUser The object or objects this property applies to.

LastUsedProjPath

String; read-only
The name of the project path that was last used.
Latitude Property

The latitude of the point.

See Also | Example

Signature

object.Latitude

object

CogoPoint The object or objects this property applies to.

Latitude

Double; read-only
LayerFile Property

The layer file to apply to the drawing.

See Also | Example

Signature

object.LayerFile

object

DatabasePreferences The object or objects this property applies to.

LayerFile

String; read-write

The name of the layer file.

Remarks

The layer file is stored as a global setting in the registry, and not on a per-drawing basis. An entry is maintained for each MeasurementUnit.
LayerStandard Property

The layer standard to apply to the drawing.

**Signature**

object.LayerStandard

object

DatabasePreferences The object or objects this property applies to.

LayerStandard

String; read-write
The name of the layer standard.

**Remarks**

The layer standards are stored on a per drawing basis in the dictionary.
LeftWidth Property (Civil Engineering Feature)

Returns the left width of the alignment cross section.

See Also | Example

Signature

object.LeftWidth

object

CrossSectionBlock The object or objects this property applies to.

LeftWidth

Double; read-only
The left width of the alignment cross section.
Length Property

The length of the alignment curve, spiral or tangent or parcel curve or line.

See Also | Example

Signature: Overview

- AlignEntity, AlignCurve, AlignSpiral, AlignTangent
- ParcelEntity, ParcelCurve, ParcelLine

The length of the alignment curve, spiral or tangent.

Signature

object.Length

object

AlignEntity, AlignCurve, AlignSpiral, AlignTangent
The object or objects this property applies to.

Length

Double; read-only
The length of the alignment curve, spiral or line.

The length of the parcel curve or line.

Signature
object.Length

object

ParcelEntity, ParcelCurve, ParcelLine
The object or objects this property applies to.

Length

Double; read-only
The length of the parcel curve or line.
**LExt Property**

Returns the external length of the spiral.

### Signature

```plaintext
object.LExt
```

- **object**
  - The object or objects this property applies to.

- **LExt**
  - Double; read-only
  - The external length of the spiral.

### Remarks

The LExt is the external length of the spiral. For a simple spiral, the value is 0.0. This represents the length from the TS to the External point in a compound spiral condition.

---

**See Also** | **Example**
LinearDisplayFormat Property

The format used to display linear values.

See Also | Example

Signature

object.LinearDisplayFormat

object

DatabasePreferences The object or objects this property applies to.

LinearDisplayFormat

Long; read-write

1:  Scientific

2:  Decimal

3:  Engineering

4:  Architectural

5:  Fractional
LinearPrecision Property

Specifies the linear precision for the drawing.

See Also | Example

Signature

object.LinearPrecision

object

DatabasePreferences The object or objects this property applies to.

LinearPrecision

Long; read-write
The number of decimals to the right of the decimal point.

Remarks

The precision settings are used only for labeling and listing values, not actual computations. AutoCADLand Desktop commands always calculate all numbers up to the highest internal precision.
**LinearUnit Property**

The unit used to display linear values.

**Signature**

```csharp
object.LinearUnit
```

**DatabasePreferences** The object or objects this property applies to.

**LinearUnits**

AecBuiltInUnit enum; read-write

- aecUnitInch: Inch units
- aecUnitFoot: Foot units
- aecUnitMillimeter: Millimeter units
- aecUnitCentimeter: Centimeter units
- aecUnitDecimeter: Decimeter units
- aecUnitMeter: Meter units
Remarks

Use this property to specify whether you want to use feet or meters in your drawing.

The linear units that you select must be the same for all drawings in a project. You cannot mix foot-based drawings with meter-based drawings in the same project.
LockedPointNumbers Property

Returns a list of the locked points in the current project point database.

See Also | Example

Signature

object.LockedPointNumbers()

object

CogoPoints The object or objects this property applies to.

LockedPointNumbers

String; read-only
Comma delimited string with all used points with groups separated by hyphens.

Remarks

Typical examples of the point string format are: "1,2,3,5,10" "1-100,1000-2000"
LockType Property

Specifies the type of lock on the Alignment, COGO Point, or Surface.

See Also | Example

Signature

object.LockType

object

Alignment, CogoPoint, FileLock, Surface
The object or objects this property applies to.

LockType

eAeccFileLockType; read-only

kNoLock: No lock exists

kReadLock: User can only read the object

kWriteLock: User can read and write the object
**LOffset Property**

The offset distance for the spiral.

**Signature**

```
object.LOffset
```

object

*AlignSpiral*  The object or objects this property applies to.

**Remarks**

If the spiral is an offset spiral (see property SpiralType2), this property is set. Otherwise, it is 0.0.
Longitude Property

The longitude of a point.

See Also | Example

Signature

object.Longitude

object

CogoPoint The object or objects this property applies to.

Longitude

Double; read-only
MaxElevation Property

Returns the Maximum Elevation of a Cross Section or Surface.

Signature: Overview

| CrossSection (Civil Engineering Feature) |
| Surface |

Returns the Maximum Elevation of a Cross Section.

Signature

RetVal = object.MaxElevation

object

CrossSection The object or objects this property applies to.

MaxElevation Double; read-only
The Maximum Elevation of the Cross Section.

Returns the Maximum Elevation of a Surface.

Signature

RetVal = object.MaxElevation
object

**Surface**
The object or objects this property applies to.

MaxElevation

Double; read-only
The Maximum Elevation of the Surface.
MaxFaceArea Property

The area of the largest Face for the Surface.

See Also | Example

Signature

object.MaxFaceArea

object

Surface The object or objects this property applies to.

MaxFaceArea

Double; read-only
The area of the largest Face.

Remarks

This is an Extended Statistic.
MaxGrade Property

The maximum grade for the Surface.

**Signature**

object.MaxGrade

object

**Surface** The object or objects this property applies to.

MaxGrade

Double; read-only
The maximum grade for the surface.

**Remarks**

This is an Extended Statistic.
MaxOffset Property (Civil Engineering Feature)

Returns the Maximum Offset of the Cross Section.

See Also | Example

Signature

RetVal = object.MaxOffset

object

CrossSection The object or objects this property applies to.

MaxOffset

Double; read-only
The Maximum Offset of the Cross Section.
MeanElevation Property

The mean elevation of the Surface.

See Also | Example

Signature

object.MeanElevation

object

Surface The object or objects this property applies to.

MeanElevation

Double; read-only
The mean elevation of the Surface.

Remarks

This is an Extended Statistic.
**MeasurementUnit Property**

The system of units used to display measurements.

See Also | Example

**Signature**

object.MeasurementUnit

object

[DatabasePreferences](#) The object or objects this property applies to.

**MeasurementUnit**

AcMeasurementUnit enum; read-only

acEnglish:  English units

acMetric:   Metric units
MidOrdinate Property

The mid ordinate for the curve.

See Also | Example

Signature

object.MidOrdinate

object

AlignCurve The object or objects this property applies to.

MidOrdinate

Double; read-only
The mid ordinate length for the curve.
MinDepressionArea Property

The minimum depression area for all watersheds.

Signature

object.MinDepressionArea

    object
        WaterSheds  The object or objects this property applies to.

    MinDepressionArea
        Double; read-only
        The minimum depression area for all watersheds.
MinDepressionDepth Property

The minimum depression depth for all watersheds.

See Also | Example

Signature

object.MinDepressionDepth

object

WaterSheds The object or objects this property applies to.

MinDepressionDepth

Double; read-only
The minimum depression depth for all watersheds.
MinElevation Property

Returns a Minimum Elevation of a Cross Section or Surface.

See Also | Example

Signature: Overview

I CrossSection (Civil Engineering Feature)

I Surface

Returns the Minimum Elevation of a Cross Section.

Signature

RetVal = object.MinElevation

object

CrossSection The object or objects this property applies to.

MinElevation

Double; read-only
The Minimum Elevation of the Cross Section.

Returns the Minimum Elevation of a Surface.

Signature

RetVal = object.MinElevation
**object**

**Surface**
The object or objects this property applies to.

**MinElevation**

Double; read-only
The Minimum Elevation of the Surface.
**MinFaceArea Property**

The area of the smallest Face for the Surface.

**Signature**

object.MinFaceArea

- **object**
  - `Surface` The object or objects this property applies to.

- **MinFaceArea**
  - Double; read-only
  - The area of the smallest Face.

**Remarks**

This is an Extended Statistic.
MinGrade Property

The minimum grade for the Surface.

Signature

object.MinGrade

object

Surface The object or objects this property applies to.

MinGrade

Double; read-only
The minimum grade for the surface.

Remarks

This is an Extended Statistic.
MinOffset Property (Civil Engineering Feature)

Returns the Minimum Offset of the Cross Section.

See Also | Example

Signature

RetVal = object.MinOffset

object

CrossSection The object or objects this property applies to.

MinOffset

Double; read-only

The Minimum Offset of the Cross Section.
<table>
<thead>
<tr>
<th>Name Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets the name of various objects.</td>
</tr>
</tbody>
</table>

**Signature: Overview**

- AeccContourStyle
- Alignment
- CogoPoint
- CrossSectionBlock (Civil Engineering Feature)
- CrossSectionSurface (Civil Engineering Feature)
- DEMFile
- DescriptionKeyFile
- Drawing
- FGProfiles (Civil Engineering Feature)
- Parcel
- PointFile
- PointGroup
- PointGroupName
The name of the contour style.

**Signature**

```
object.Name

object

*AeccContourStyle* The object or objects this property applies to.
```

Name

String; read-only
The name of the contour style.

The name of the alignment.

**Signature**

```
object.Name

object

*Alignment* The object or objects this property applies to.
```

Name

String; read-only
The name of the alignment.

The point name.
**Signature**

`object.Name`

`object`

**CogoPoint**

The object or objects this property applies to.

**Name**

String; read-only

**Remarks**

When the project is created, an option exists to create Cogo points with Names. If the project does not use point names, then this property will be empty and can not be set.

Returns the name of the cross sections alignment profile.

**Signature**

`object.Name`

`object`

**CrossSectionBlock**

The object or objects this property applies to.

**Name**

String; read-only

The name of the cross sections alignment profile.

The cross section surface name.

**Signature**

`object.Name`

`object`
**CrossSectionSurface**
The object or objects this property applies to.

Name

String; read-only
The cross section surface name.

The name of the Digital Elevation Model file to be included in the Surface definition.

**Signature**

object.Name

object

**DEMFile**;
The object or objects this property applies to.

Name

String; read-only
The fully qualified name and path of the input file.

The Name for the description key file.

**Signature**

object.Name

object

**DescriptionKeyFile**;
The object or objects this property applies to.

Name

String; read-write
The name for the description key file, not including the path.
Gets the name of the drawing, not including the path.

**Signature**

`object.Name`

`object`

**Drawing**
The object or objects this property applies to.

`Name`

`String`; read-only
The name of the drawing.

Returns the Finished Grade Profiles alignment name.

**Signature**

`Object.Name`

`object`

**FGProfiles**
The object or objects this property applies to.

`Name`

`String`; read-only
The name of the Finished Grade Profile alignment.

The name of the parcel.

**Signature**

`object.Name`

`object`

**Parcel**
The object or objects this property applies to.
Name

String; read-only
The name of the parcel.

The name of the file to be included in the Surface definition.

**Signature**

object.Name

object

**PointFile**
The object or objects this property applies to.

Name

String; read-only
The name of the input file.

Specifies the Name override for the project database point group.

**Signature**

object.Name

object

**PointGroup**
The object or objects this property applies to.

Name

String; read-write

**Remarks**

If the Name property is set, then the **NameXDRef** property is cleared.

The Name of the PointGroup to be used to build the Surface.
**Signature**

object.Name

object

*PointGroupName*

The object or objects this property applies to.

Name

String; read-only

Name of the PointGroup.

**Remarks**

The PointGroupName is a link between Cogo Point Groups and TM. By using a PointGroupName, that Cogo Point Group will be included in the Surface definition.

No validation exists between a PointGroupName and a Cogo Point Group. If the PointGroupName.Name does not exist as a Cogo Point Group, no errors will be generated.

Returns the Alignment name for the Profile.

**Signature**

Object.Name

object

*ProfileBlock*

The object or objects this property applies to.

Name

String; read-write

The Alignment name for the Profile.

Specifies the name of the project.
**Signature**

object.Name

**object**

*Project*
The object or objects this property applies to.

**Name**

String; read-only
The name of the project.

**Remarks**

This property returns the directory name only, not the **Path**.

The name of the prototype.

**Signature**

object.Name

**object**

*Prototype*
The object or objects this property applies to.

**Name**

String; read-write
The name of the prototype.

**Remarks**

Use this property to rename a prototype.

The Name of the Surface.

**Signature**
object.Name

object

**Surface**
The object or objects this property applies to.

Name

String; read-only
The Name of the Surface.

**Remarks**

Surface names are limited to 40 characters.
NameOverride Property

Specifies if the name for the project database point will be overridden from the Point Group.

See Also | Example

Signature

object.NameOverride

object

PointGroup The object or objects this property applies to.

NameOverride

Boolean; read-write
TRUE: The Cogo project point name will be set by the Point Group.
FALSE: The Cogo project point name will not be set by the Point Group.

Remarks

When you set this value to FALSE, the Name property will be cleared.
NameXDRef Property

Specifies the Name override for the project database point group. The name will be taken from the external database reference (XDRef).

See Also | Example

Signature

object.NameXDRef

object

PointGroup The object or objects this property applies to.

NameXDRef

String; read-write
The name of the external database reference.

Remarks

If the NameXDRef property is set, then the Name property is cleared.
**NextPointNumber Property**

Gets / sets the next sequential point number for point creation.

**Signature**

object.NextPointNumber

    object
        CogoPoints The object or objects this property applies to.
    NextPointNumber
        Long; read-write

**Remarks**

The default number when you begin a project is 1. An error is returned if you try to set the property to a value less than 1 or a used point number.
Normal Property

Returns the Normal for the Face.

See Also | Example

Signature

object.Normal

object

Face The object or objects this property applies to.

Normal

Variant (3 element array of doubles); read-only

Returns the normal for the Face.
Northing Property

Gets the Northing coordinates of various objects.

See Also | Example

Signature: Overview

- **AeccPoint**
- **CogoPoint**
- **TinPoint**

The Northing coordinate of the point.

**Signature**

```csharp
object.Northing

object

**AeccPoint** The object or objects this property applies to.

Northing

Double; read-write
The Northing coordinate of the point.

Remarks

The Easting coordinate is derived from the WCS X coordinate with the **BasePoint** and **NorthRotation** applied.
The north coordinate for the point.

**Signature**

object.Northing

- object
- **CogoPoint**
  The object or objects this property applies to.

Northing

- Double; read-write

The Northing coordinate for the TinPoint.

**Signature**

object.Northing

- object
- **TinPoint**
  The object or objects this property applies to.

Northing

- Double; read-only
  The Northing coordinate.
NorthRotation Property

Specifies the north rotation for your drawing layout.

See Also | Example

Signature

object.NorthRotation

object

DatabasePreferences The object or objects this property applies to.

NorthRotation

Double; read-write
The north rotation angle. The angle is in radians and measured counter-clockwise from the X axis (East).

Remarks

The initial value for this property is 0.

If you change the NorthRotation in an existing drawing then you must move any objects in the drawing to match the new coordinate system.
Number Property

The number property for various objects.

Signature: Overview

- **AeccPoint**
- **Alignment**
- **CogoPoint**
- **Parcel**

The AeccPoint point number.

**Signature**

```plaintext
object.Number

object

AeccPoint

The object or objects this property applies to.

Number

Long; read-only
The point number.
```

**Remarks**
The number property must be unique if the AeccPoint is to be imported into the Cogo database.

The Alignment number.

**Signature**

object.Number

object

Alignment
The object or objects this property applies to.

Number
Long; read-only
The number of the alignment.

The CogoPoint number.

**Signature**

object.Number

object

CogoPoint
The object or objects this property applies to.

Number
Long; read-only

**Remarks**

The point number is set with the value of the NextPointNumber property of the CogoPoints collection when the Add method is invoked, and cannot be changed.

The number of the Parcel.
**Signature**

**object.**Number

**object**

**Parcel**
The object or objects this property applies to.

**Number**

Long; read-only
The number of the parcel.
NumberOfFaces Property

The number of Faces for the Surface.

Signature

object.NumberOfFaces

object

Surface The object or objects this property applies to.

NumberOfFaces

Long; read-only

The number of faces in the surface.

Remarks

This is an Extended Statistic.
NumberOfPoints Property

The number of TinPoints for the Surface.

See Also | Example

Signature

object.NumberOfPoints

object

Surface The object or objects this property applies to.

NumberOfPoints

Long; read-only
The number of points in the surface.
ObjectID Property

Returns the Object ID for the entity.

See Also | Example

Signature

object.ObjectID

object

AlignCurve, AlignEntity, AlignSpiral, AlignTangent
The object this property applies to.

ObjectID

Long; read-only
The Object ID for the entity.

Remarks

An Alignment can exist in multiple drawings. The Object ID for the entity is for the current drawing only.

Valid alignment ObjectIDs are defined during an Import. The ObjectIDs are valid until the alignment is saved. Alignments are automatically saved each time an alignment property is modified when AutoSave is set to true, or when an alignment Save is executed.
Offset Property (Civil Engineering Feature)

Returns the cross section point code offset.

See Also | Example

Signature

object.Offset

  object

  CrossSectionPointCode The object or objects this property applies to.

Offset

  Double; read-only
  The cross section point code offset.
OffsetElevations Property (Civil Engineering Feature)

Returns the cross section surface offset and elevations.

See Also | Example

Signature

object.OffsetElevations

object

CrossSectionSurface The object or objects this property applies to.

OffsetElevations

Variant (array of doubles); read-only
Returns the cross section surface offset and elevations.
Outputs Property

Returns the SurfaceOutputs object for the surface.

See Also | Example

Signature

object.Outputs

object

Surface The object or objects this property applies to.

Inputs

SurfaceOutputs; read-only
All outputs for the surface.
OverflowPoints Property

Returns the WaterShed overflow points.

**Signature**

object.OverflowPoints

object

[WaterShed](#) The object or objects this property applies to.

OverflowPoints

Variant (3-element array of doubles); read-only
The points that make up the calculated watershed.

See Also | Example
OverrideDescription Property

Returns the description that has been overridden by the PointGroup.

See Also | Example

Signature

object.OverrideDescription

object

CogoPoint The object or objects this property applies to.

OverrideDescription

String; read-only

Remarks

If the GroupName has been set for the CogoPoints collection and that PointGroup has the description override, this value will be set.
OverrideElevation Property

Returns the elevation that has been overridden by the PointGroup.

See Also | Example

Signature

object.OverrideElevation

object

CogoPoint The object or objects this property applies to.

OverrideElevation

Double; read-only

Remarks

If the GroupName has been set for the CogoPoints collection and that PointGroup has the elevation override, this value will be set.
OverrideName Property

Returns the point name that has been overridden by the PointGroup.

**Signature**

object.OverrideName

  object
  
  CogoPoint The object or objects this property applies to.

  OverrideName
  
  String; read-only

**Remarks**

If the GroupName has been set for the CogoPoints collection and that PointGroup has the name override, this value will be set.

See Also | Example
OverrideNew Property

Toggles the display of the AutoCAD Land Desktop New dialog box.

**Signature**

object.OverrideNew

  object  
  PreferencesUser The object or objects this property applies to.

  OverrideNew  
  Boolean; read-write  
  TRUE: Displays the AutoCAD Land Desktop New dialog box.  
  FALSE: Displays the AutoCAD New dialog box.

**Remarks**

The initial value for this property is TRUE.

If you set this property to FALSE and use the AutoCAD New command to create a new drawing, then you must save the new drawing to name it. Only named drawings can be associated with a project.

You must exit AutoCAD Land Desktop and restart for this change to take effect.
OverrideOpen Property

Toggles the display of the AutoCAD Land Desktop Open dialog box.

See Also | Example

Signature

object.OverrideOpen

object

PreferencesUser The object or objects this property applies to.

OverrideOpen

Boolean; read-write
TRUE: Displays the AutoCAD Land Desktop Open dialog box.
FALSE: Displays the AutoCAD open dialog box.

Remarks

The initial value for this property is TRUE.

If you set this property to FALSE (and use the AutoCAD Open command to open a drawing), then you will automatically be prompted to select a project if the drawing isn’t associated with a project or if the drawing’s project is not found.

You must exit AutoCAD Land Desktop and restart for this change to take effect.
**Owner Property**

The owner (AutoCAD login name) of the lock owner.

**Signature**

object.Owner

object

[FileLock](#) The object or objects this property applies to.

Owner

String; read-only

The owner of the file lock.
P Property

The spiral's P value.

See Also | Example

Signature

object.P

object

AlignSpiral The object or objects this property applies to.

P

Double; read-only
The spiral's P value.
Parcel Property

The collection of parcel preferences for the project.

See Also | Example

Signature

object.Parcel

object

PreferencesProject The object or objects this property applies to.

Parcel

PreferencesParcel; read-only
The PreferencesParcel object.
**ParcelEntities Property**

The collection of geometric entities for the parcel.

**Signature**

object.ParcelEntities

object  Parcel  The object or objects this property applies to.

ParcelEntities  ParcelEntities collection; read-only
The ParcelEntities collection.
Parcels Property

Gets the Parcels collection.

See Also | Example

Signature

object.Parcels

object

Project The object or objects this property applies to.

Parcels

Parcels collection; read-only
The Parcels collection.

Remarks

The Parcels collection represents all of the Parcel objects in this project. This property is valid only for the ActiveProject.
Path Property

Gets the path of various objects.

**Signature: Overview**

- `DescriptionKeyFile`
- `DescriptionKeyFiles`
- `Drawing`
- `Drawings`
- `FileLocks`
- `Project`

The path for the description key files for a project.

**Signature**

```
object.Path

object DescriptionKeyFile, DescriptionKeyFiles
The object or objects this property applies to.

Path
String; read-only
```
Gets the path of the drawing.

**Signature**

object.Path

object

**Drawing**
The object or objects this property applies to.

Path

String; read-only
The path of the drawing.

**Remarks**

This value is read-only and cannot be changed.

_________________________________________________________________________________________________________________

Returns the path at which the drawings for this project are located.

**Signature**

object.Path

object

**Drawings**
The object or objects this property applies to.

Path

String; read-only
The keyword for the project.

_________________________________________________________________________________________________________________

Specifies the path at which the project locks are located.

**Signature**

object.Path
object

**FileLocks**
The object or objects this property applies to.

Path

String; read-only
The path name to for the collection.

**Remarks**
Use this property to find locks for a given project.

---

Specifies the path of the project.

**Signature**

object.Path

object

**Project**
The object or objects this property applies to.

Path

String; read-only
The path of the project.

**Remarks**
The path of the project contains the **Name** as the last folder in the path specification.
Perimeter Property

The perimeter of the parcel.

**Signature**

object.Perimeter

object

**Parcel** The object or objects this property applies to.

Perimeter

Double; read-only
The perimeter of the parcel.
**PiEasting Property**

The Easting coordinate for the curve's PI.

**Signature**

`object.PiEasting`

- `object`
  - `AlignCurve` The object or objects this property applies to.

- `PiEasting`
  - `Double; read-only`
  - The east coordinate for the PI of the curve.

[See Also](#) | [Example](#)
PiNorthing Property

The Northing coordinate for the curve's PI.

See Also | Example

Signature

object.PiNorthing

object

    AlignCurve The object or objects this property applies to.

PiNorthing

    Double; read-only
    The north coordinate for the PI of the curve.
PointFiles Property

Returns the PointFiles collection.

See Also | Example

Signature

object.PointFiles

object

SurfaceInputs The object or objects this property applies to.

PointFiles

PointFiles collection; read-only
The PointFiles collection.
PointGroupNames Property

Returns the PointGroupNames collection.

**Signature**

`object.PointGroupNames`

- `object` *SurfaceInputs* The object or objects this property applies to.

- `PointGroupNames` *PointGroupNames* collection; read-only
  The *PointGroupNames* collection.

---

See Also | Example
## PointGroups Property

Gets the PointGroups collection.

### Signature

```
object.PointGroups
```

- **object**
  - `Project` The object or objects this property applies to.

- **PointGroups**
  - `PointGroups` collection; read-only
  - The `PointGroups` collection.

### Remarks

The `PointGroups` collection represents all of the `PointGroup` objects in this project. This property is valid only for the `ActiveProject`.
**PointList Property**

Specifies the list of project database points for the point group.

**Signature**

object.PointList

- **object**
  - [PointGroup](#) The object or objects this property applies to.

- **PointList**
  - String; read-write
  - Comma delimited string with all used points with groups separated by hyphens.

**Remarks**

Examples: "1,2,3,5,10"
"1-100,1000-2000"
**PointNameSize Property**

Returns maximum number of characters supported in point name.

**Signature**

object.PointNameSize

object

[CogoPoints](#) The object or objects this property applies to.

**NextPointNumber**

Long; read-only

Returns a long that represents the maximum number of characters that a point name can hold.
**PointOnLineTolerance Property**

Returns the minimum distance between a line and a point not on the line.

See Also | Example

**Signature**

object.PointOnLineTolerance

object

[Surface](#) The object or objects this property applies to.

PointOnLineTolerance

Double; read-only
The point on line tolerance for the surface.
PointTolerance Property

Returns the minimum distance between distinct points.

See Also | Example

**Signature**

object.PointTolerance

object

Surface The object or objects this property applies to.

PointTolerance

Double; read-only

The point tolerance for the surface.
**Precision Property**

The precision used to truncate area labels when a Parcel is imported to the drawing.

**Signature**

object.Precision

```object```

**Parcel** The object or objects this property applies to.

**Precision**

Double; read-only
The parcel's Precision value.
Preferences Property

The preferences property of various objects.

See Also | Example

Signature: Overview

I AeccApplication

I Document

I Project

Get the AeccApplication Preferences

Signature

object.Preferences

object

AeccApplication The object or objects this property applies to.

Preferences

Preferences object; read-only

Remarks

The Preferences object holds the system level options.
Get the Preferences object.

**Signature**

object.Preferences

object

**Document**
The object or objects this property applies to.

Preferences

**DatabasePreferences** object; read-only
The **DatabasePreferences** object.

**Remarks**

The **DatabasePreferences** object holds the settings that are saved in the AutoCAD Land Desktop drawing.

Get the PreferencesProject object.

**Signature**

object.Preferences

object

**Project**
The object or objects this property applies to.

Preferences

**PreferencesProject** object; read-only
The **PreferencesProject** object.

**Remarks**

The Preferences object holds the project level options. This property is valid only for the **ActiveProject**.
PreferencesPath Property

Specifies the directory in which the preference settings are located.

See Also | Example

Signature

object.PreferencesPath

object

PreferencesFiles The object or objects this property applies to.

PreferencePath

String; read-write
The drive and path for the preferences.

Remarks

The preference settings are stored in the C:\Documents and Settings\All Users\Application Data\Autodesk\AutoCAD Land Desktop <version number>\<release number>\Data\pref folder. The file name is <AutoCAD login name>.dfm. The preference path settings are stored in the sdsk.dfm file in the program folder.

You must exit AutoCAD Land Desktop and restart for this change to take effect.
Profile Property (Civil Engineering Feature)

The collection of vertical profile preferences for the project.

See Also | Example

**Signature**

object.Profile

  object

  PreferencesProject The object or objects this property applies to.

  PreferencesProfile; read-only
  The PreferencesProfile object.
ProfileBlocks Property

Gets the ProfileBlocks object.

See Also | Example

Signature

object.ProfileBlock

object

Document The object or objects this property applies to.

ProfileBlocks

ProfileBlocks object; read-only

The ProfileBlocks object.
ProgramPath Property

Specifies the directory in which the AutoCAD Land Desktop is installed.

See Also | Example

Signature

object.ProgramPath

object

PreferencesFiles The object or objects this property applies to.

ProgramPath

String; read-only
The drive and path where the AutoCAD Land Desktop is installed.
**ProjectName Property**

The name of the project that the drawing belongs to.

**Signature**

`object.ProjectName`

`object DatabasePreferences` The object or objects this property applies to.

**Remarks**

The value must 255 characters or less in length.
ProjectPath Property

Gets the project path of various objects.

See Also | Example

Signature: Overview

PreferencesFiles
Projects

Specifies the directory in which the projects are located.

Signature

object.ProjectPath

object
PreferencesFiles The object or objects this property applies to.

ProjectPath

String; read-write
The drive and path for projects.

Remarks

By default, the project path is c:\Land Projects. You must exit AutoCAD Land Desktop and restart for this change to take effect.
Specifies the directory in which the projects are located.

**Signature**

object.ProjectPath

object

Projects
The object or objects this property applies to.

ProjectPath
String; read-write
The project path.

**Remarks**

The initial value of this property is the same as the ProjectPath property of the PreferencesFiles object.

The members of this collection are AutoCAD Land Desktop projects located at this path.
Projects Property

Gets the Projects collection.

See Also | Example

Signature

object.Properties

object

AeccApplication The object or objects this property applies to.

Projects

Projects collection; read-only

Remarks

The Projects collection allows you to access all the AutoCAD Land Desktop projects available on the network. This includes all project data stored outside the drawings.
PrototypeName Property

Specifies the name of the prototype used for the project.

**Signature**

object.PrototypeName

object

PrototypeName

String; read-only

The name of the prototype used for this project.
PrototypePath Property

Specifies the directory in which the project prototype files are located.

See Also | Example

Signature

object.prototypePath

object

PreferencesFiles The object or objects this property applies to.

PrototypePath

String; read-write
The drive and path for the project prototype files.

Remarks

By default, this path is C:\Documents and Settings\All Users\Application Data\Autodesk\AutoCAD Land Desktop <version number>\<release number>\Data\Prototypes. You must exit AutoCAD Land Desktop and restart for this change to take effect.
Prototypes Property

Gets the Prototypes collection.

See Also | Example

Signature

object.Prototypes

object

AeccApplication

The object or objects this property applies to.

Prototypes

Prototypes collection; read-only

Remarks

The Prototypes collection allows you to access all the prototypes available for configuring your next AutoCAD Land Desktop project.
PVIs Property (Civil Engineering Feature)

Gets the PVIs collection.

**Signature**

object.PVIs

object

[FGProfile](#) The object or objects this property applies to.

PVIs

[PVIs](#) collection; read-only
The PVIs collection.
RadialDistance Property

The radial distance of the spiral.

See Also | Example

Signature

object.RadialDistance

object AlignSpiral The object or objects this property applies to.

RadialDistance

Double; read-only
The radial distance at the SC.
Radius Property

The radius of the alignment or parcel curve.

See Also | Example

Signature: Overview

I AlignCurve

I ParcelCurve

The radius of the alignment curve.

Signature

object.Radius

object

AlignCurve The object or objects this property applies to.

Radius

Double; read-only
The radius of the alignment curve.

The radius of the parcel curve.

Signature

object.Radius
object

**ParcelCurve**
The object or objects this property applies to.

Radius

Double; read-only
The radius of the parcel curve.
**RawDescription Property**

The raw description for the COGO project database point.

See Also | Example

**Signature**

object.RawDescription

    object

    **CogoPoint** The object or objects this property applies to.

    RawDescription

    String; read-write

**Remarks**

The RawDescription for the Cogo point does not include PointGroup Overrides or DescriptionKeyFile. If a Description Key is used, then it is based on the RawDescription.
RevisionNumber Property

The revision number for the Surface.

See Also | Example

Signature

object.RevisionNumber

object

Surface The object or objects this property applies to.

RevisionNumber

Long; read-only
The surface revision number.

Remarks

Each time the Surface is built, the revision number is updated.
RightWidth Property (Civil Engineering Feature)

Returns the right width of the alignment cross section.

See Also | Example

Signature

object.RightWidth

object

CrossSectionBlock The object or objects this property applies to.

RightWidth

Double; read-only

The right width of the alignment cross section.
**RotateByDescriptionParam Property**

If TRUE, only the RotateDescriptionParam is rotated.

**Signature**

object.RotateByDescriptionParam

object

**DescriptionKey** The object or objects this property applies to.

RotateByDescriptionParam

Boolean; read-write

TRUE: Only the RotateDescriptionParam is rotated.

FALSE: Do not rotate by RotateDescriptionParam only.

See Also | Example
RotateByFixedFactor Property

If TRUE, rotate by the RotateFactor.

Signature

object.RotateByFixedFactor

  object

  DescriptionKey The object or objects this property applies to.

  RotateByFixedFactor

  Boolean; read-write
  TRUE: Rotate by the RotateFixedFactor.
  FALSE: Do not rotate by RotateFixedFactor only.
RotateClockwise Property

The rotation direction.

Signature

object.RotateClockwise

object

DescriptionKey The object or objects this property applies to.

RotateClockwise

Double; read-write
The rotate direction.
RotateDescriptionParam Property

The description parameter for rotation.

See Also | Example

Signature

object.RotateDescriptionParam

object

  DescriptionKey The object or objects this property applies to.

  RotateDescriptionParam

    Integer; read-write
    Returns and sets the rotate description parameter.
RotateFixedFactor Property

The rotation factor.

Signature

object.RotateFixedFactor

object

DescriptionKey The object or objects this property applies to.

RotateFixedFactor

Double; read-write
The rotate fixed factor.
ScaleByDescriptionParam Property

If TRUE, only the ScaleDescriptionParam is scaled.

See Also | Example

Signature

object.ScaleByDescriptionParam

object

DescriptionKey The object or objects this property applies to.

ScaleByDescriptionParam

Boolean; read-write
TRUE: Only the ScaleDescriptionParam is scaled.
FALSE: Not scaled by the ScaleDescriptionParam
**ScaleByDrawingScale Property**

If TRUE, the drawing scale factor is applied.

**Signature**

```plaintext
object.ScaleByDrawingScale
```

- `object`
  - *DescriptionKey* The object or objects this property applies to.

- `ScaleByDrawingScale`
  - Boolean; read-write
  - TRUE: Scale by the drawing scale factor.
  - FALSE: Do not scale by drawing scale factor.
ScaleByFixedFactor Property

If TRUE, the fixed scale factor ScaleFactor is applied.

See Also | Example

**Signature**

object.ScaleByFixedFactor

object

DescriptionKey The object or objects this property applies to.

ScaleByFixedFactor

Boolean; read-write
TRUE: Scale by the fixed scale factor ScaleFixedFactor.
FALSE: Do not scale by fixed scale factor ScaleFixedFactor.
ScaleDescriptionParam Property

The description parameter for scaling.

See Also | Example

Signature

object.ScaleDescriptionParam

object

DescriptionKey The object or objects this property applies to.

ScaleDescriptionParam

Integer; read-write

Returns and sets the scale description parameter.
ScaleFixedFactor Property

The fixed scale factor.

See Also | Example

Signature

object.ScaleFixedFactor

object

DescriptionKey

The object or objects this property applies to.

ScaleFixedFactor

Double; read-write
ScaleInXY Property

If TRUE, the symbol is scaled in the XY plane

See Also | Example

Signature

object.ScaleXY

object

DescriptionKey The object or objects this property applies to.

ScaleXY

Boolean; read-write
TRUE: Symbol is scaled in the XY plane.
FALSE: Do not scale in the XY plane.
ScaleInZ Property

If TRUE, the symbol is scaled in the Z axis.

See Also | Example

Signature

object.ScaleInZ

object

DescriptionKey The object or objects this property applies to.

ScaleInZ

Boolean; read-write
TRUE: Symbol is scaled in the Z axis.
FALSE: Do not scale in the Z axis.
### ScaleOnInsert Property

Specifies whether objects are automatically scaled when inserted into the drawing.

#### Signature

`object.ScaleOnInsert`

- `object` - The object or objects this property applies to.
- `ScaleOnInsert` - Boolean; read-write
  - TRUE: New objects are scaled automatically.
  - FALSE: New objects are not scaled automatically.
SearchType Property

Returns how the Faces collection was generated.

See Also | Example

Signature

object.SearchType

object

Faces The object or objects this property applies to.

SearchType

eAeccSurfaceSearchType enum; read-only

kNoSearch: No filter has been applied

kSearchByPath: Search by path filter

kSearchByBoundary: Search by surface boundary

kSearchByPoint: Search by point filter

Remarks
The SearchType property describes how the Faces collection was generated. The filter returns only the faces that exist between two points.
SheetHeight Property

Specifies the height of the sheet.

See Also | Example

Signature

object.SheetHeight

object

DatabasePreferences The object or objects this property applies to.

SheetHeight

Double; read-write

The sheet height.

Remarks

The SheetHeight, SheetWidth and DatabaseScale determine the effective area on the drawing in relation to the plotted drawing.
SheetWidth Property

Specifies the width of the sheet.

See Also | Example

Signature

object.SheetWidth

object

DatabasePreferences The object or objects this property applies to.

SheetWidth

Double; read-write
The sheet width.

Remarks

The SheetHeight, SheetWidth and DatabaseScale determine the effective area on the drawing in relation to the plotted drawing.
**ShortTangent Property**

Returns the short tangent length for compound and simple spirals.

**Signature**

object.ShortTangent

  object

    AlignSpiral The object or objects this property applies to.

  ShortTangent

    Double; read-only
    The short tangent length.
ShowStartupDialog Property

Toggles the display of the startup dialog.

See Also | Example

Signature

object.ShowStartupDialog

object

PreferencesUser The object or objects this property applies to.

ShowStartupDialog

Boolean; read-write

TRUE: Displays the startup dialog.
FALSE: Hides the startup dialog.

Remarks

The initial value for this property is TRUE. If this property is set to FALSE, the OverrideNew and OverrideOpen properties are also set to FALSE.

You must exit AutoCAD Land Desktop and restart for this change to take effect.
ShowSubfolders Property

Determines whether drawings in subfolders are included.

See Also | Example

Signature

object.ShowSubfolders

object

Drawings The object or objects this property applies to.

ShowSubfolders Boolean; read-write
TRUE: Drawings in subfolders are included in the collection.
FALSE: Drawings in subfolders are not included in the collection.

Remarks

The Preferences object holds the project level options.
SpeedTablesPath Property

Specifies the directory in which the speed tables are located.

See Also | Example

Signature

object.SpeedTablesPath

object

PreferencesFiles The object or objects this property applies to.

SpeedTablesPath

String; read-write
The drive and path for the speed table files.

Remarks

By default, this path is C:\Documents and Settings\All Users\Application Data\Autodesk\AutoCAD Land Desktop <version number>\<release number>\Data\Speed Tables. You must exit AutoCAD Land Desktop and restart for this change to take effect.
**SpiEasting Property**

The Easting coordinate for the spi point.

**Signature**

```plaintext
object.SpiEasting
```

object

*AlignSpiral* The object or objects this property applies to.

SpiEasting

```plaintext
Double; read-only
```

The Easting coordinate for the Spiral Point of Intersection.
SpilTangent Property

Returns the long tangent length.

See Also | Example

Signature

object.SpilTangent

object

AlignSpiral The object or objects this property applies to.

SpilTangent

Double; read-only
The long tangent length.
**SpiNorthing Property**

The Northing coordinate for the spi point.

**Signature**

object.SpiNorthing

object

[AlignSpiral] The object or objects this property applies to.

SpiNorthing

Double; read-only
The Northing coordinate for the Spiral Point of Intersection.
SpiralType1 Property

Returns the calculation method of the spiral.

**Signature**

object.SpiralType1

object

**AlignSpiral** The object or objects this property applies to.

SpiralType1
eAeccSpiralType1 enum; read-only

kClothoid: Clothoid spiral

kSinusoid: Sinusoid spiral

kCosinusoid: Cosinusoid spiral

kQuadratic: Quadratic spiral

**Remarks**

Compound spirals are only supported when using a Clothoid spiral type. No matter what spiral type is set, the command always uses the Clothoid spiral type.
SpiralType2 Property

Returns whether the spiral is a simple, offset, or compound spiral.

See Also | Example

Signature

object.SpiralType2

object

AlignSpiral The object or objects this property applies to.

SpiralType2

eAeccSpiralType2 enum; read-only

kSimple: Simple spiral

kOffset: Offset spiral

kCompound: Compound spiral

kOffsetCompound: Offset / compound spiral

Remarks

Compound spirals are only supported when using a Clothoid spiral type. No matter what spiral type is set, the command always uses the Clothoid spiral type.
StartDirection Property

The starting direction of the curve or spiral.

See Also | Example

Signature: Overview

1 AlignCurve

1 AlignSpiral

The starting direction of the curve.

Signature

object.StartDirection

object    

AlignCurve The object or objects this property applies to.

StartDirection    Double; read-only

The start direction of the curve.

Remarks

The direction is in radians and measured counter-clockwise from the X axis (East).
The starting direction of the spiral.

**Signature**

object.StartDirection

object

**AlignSpiral**
The object or objects this property applies to.

**StartDirection**

Double; read-only
The direction taken from the TS to the Spi. The angle is in radians and measured counter-clockwise from the X axis (East).

**Remarks**

The direction is measured in radians.
StartEasting Property

The Easting coordinate for the beginning of the individual alignment or profile entities.

**Signature: Overview**

- AlignEntity, AlignCurve, AlignSpiral, AlignTangent
- ParcelEntity, ParcelCurve, ParcelLine

The Easting coordinate for the beginning of the individual alignment entities.

**Signature**

object.StartEasting

object

AlignEntity, AlignCurve, AlignSpiral, AlignTangent

The object or objects this property applies to.

StartEasting

Double; read-only

The Easting coordinate for the beginning point.

The Easting coordinate for the beginning of the individual parcel entities.
**Signature**

object.StartEastIng

object

ParcelEntity, ParcelCurve, ParcelLine
The object or objects this property applies to.

StartEastIng

Double; read-only
The Easting coordinate for the beginning point.
StartingStation Property

The object's starting station.

See Also | Example

Signature: Overview

| AlignEntity, AlignCurve, AlignSpiral, AlignTangent |
| Alignement |
| ProfileBlock (Civil Engineering Feature) |

The StartingStation for individual entities in an alignment.

Signature

object.StartingStation

object

| AlignEntity, AlignCurve, AlignSpiral, AlignTangent |

The object or objects this property applies to.

StartingStation

Double; read-only

The starting station for the object.

The StartingStation for the overall alignment.

Signature
object.StartingStation

object

Alignment
The object or objects this property applies to.

StartingStation
Double; read-write
The starting station for the object.

Returns the StartingStation for the Profile.

Signature

object.StartingStation

object

ProfileBlock
The object or objects this property applies to.

StartingStation
Double; read-only The starting station for the Profile.
StartNorthing Property

The Northing coordinate for the beginning of the individual alignment or parcel entities.

See Also | Example

Signature: Overview

1 AlignEntity, AlignCurve, AlignSpiral, AlignTangent
1 ParcelEntity, ParcelCurve, ParcelLine

The Northing coordinate for the beginning of the individual alignment entities.

Signature

object.StartNorthing

object AlignEntity, AlignCurve, AlignSpiral, AlignTangent
The object or objects this property applies to.

BeginNorthing

Double; read-only
The Northing coordinate for the beginning point.

The Northing coordinate for the beginning of the individual parcel entities.
**Signature**

object.StartNorthing

object

**ParcelEntity, ParcelCurve, ParcelLine**
The object or objects this property applies to.

**BeginNorthing**

Double; read-only
The Northing coordinate for the beginning point.
Station Property (Civil Engineering Feature)

Returns the station for the cross section, cross section block, superelevation, or PVI.

See Also | Example

Signature: Overview

- CrossSection (Civil Engineering Feature)
- CrossSectionBlock (Civil Engineering Feature)
- PVI (Civil Engineering Feature)
- Superelevation (Civil Engineering Feature)

Returns the station of the cross section.

Signature

object.Station

object

CrossSection The object or objects this property applies to.

Station

Double; read-only

The station for the cross section.
Returns the station of the cross section block.

**Signature**

object.Station

object **CrossSectionBlock**
The object or objects this property applies to.

**Station**

Double; read-only
The station for the cross section block.

---

Returns the Station of the PVI.

**Signature**

object.Station

object **PVI**
The object or objects this property applies to.

**Station**

Double; read-only
The station for the PVI.

---

Returns the superelevation station.

**Signature**

object.Station

object **Superelevation**
The object or objects this property applies to.
Station

Double; read-only
The station for the superelevation.
StationAhead Property

The ahead station for the station equation.

Signature

object.StationAhead

object

StationEquation The object or objects this property applies to.

StationAhead

Double; read-write
The ahead station for the station equation.
StationBack Property

The back station for the station equation.

See Also | Example

Signature

object.StationBack

object

StationEquation The object or objects this property applies to.

StationBack

Double; read-write

The back station for the station equation.
StationEquations Property

The alignment's station equations collection.

**Signature**

object.StationEquations

  object  
  
  **Alignment** The object or objects this property applies to.

  StationEquations
  
  **StationEquations** collection; read-only
  The alignment station equations collection.
StationElevations Property (Civil Engineering Feature)

An array of stations and elevations for the EGProfile.

See Also | Example

Signature

object.StationElevations

object

EGProfile The object or objects this property applies to.

StationElevations

Variant (array of doubles); read-write
An array of doubles with the format of Station, Elevation, Station, etc.
StationIncrement Property (Civil Engineering Feature)

Returns the horizontal distance between vertical grid lines.

See Also | Example

**Signature**

`object.StationIncrement`

**object**

`ProfileBlock` The object or objects this property applies to.

**StationIncrement**

`Double; read-only`

The Station Increment for the Profile.
Status Property

The status for a surface.

See Also | Example

Signature

object.Status

object

Surface The object or objects this property applies to.

Status

eAeccSurfaceStatus enum; read-only

kNoData: Surface has no data

kNotBuilt: Surface is not built

kUpToDate: Surface is up to date

kOutOfDate: Surface is out of date

Remarks

The status of a surface will only be accurate when the time between a Build and an addition of Surface Input data is greater than the resolution of the file system.
SuperelevationCode Property (Civil Engineering Feature)

Returns the superelevation code.

See Also | Example

Signature

object.SuperelevationCode

object

Superelevation

The object or objects this property applies to.

SuperelevationCode

eAeccSectionSECode enum; read-only

kSEFullCrown: Full crown superelevation

kSEHalfCrown: Half crown superelevation

kSECrownRemoved: Crown removed

kSEFullSuperelevation: Full superelevation

kSEReverseCurve: Reversese curve

kSECompoundCurve: Compound curve
Superelevations Property (Civil Engineering Feature)

Gets the alignment superelevations collection.

**Signature**

object.Superelevations

  object  
  
  **Alignment** The object or objects this property applies to.

  Superelevations  
  
  **SuperElevations** collection; read-only
  The **SuperElevations** collection.
Surface Property

The collection of surface preferences for the project.

See Also | Example

**Signature**

object.Surface

object

`PreferencesProject` The object or objects this property applies to.

Surface

`PreferencesSurface`; read-only
The `PreferencesSurface` object.
SurfaceName Property (Civil Engineering Feature)

The name of the EGProfile surface.

See Also | Example

Signature

object.SurfaceName

object

EGProfile The object or objects this property applies to.

SurfaceName

String; read-only
The name of the EGProfile surface.
**Surfaces Property**

Gets the Surfaces collection.

**Signature**

object.Surfaces

object

[Project] The object or objects this property applies to.

Surfaces

[Surfaces] collection; read-only

The [Surfaces] collection.

**Remarks**

The [Surfaces] collection represents all of the [Surface] objects in this project. This property is valid only for the [ActiveProject].
SymbolBlock Property

Specifies the symbol block.

See Also | Example

**Signature**

object.SymbolBlock

- object
  - [DescriptionKey](#) The object or objects this property applies to.
- SymbolBlock
  - String; read-write
  - Returns and sets the symbol block.
SymbolLayer Property

Specifies the symbol block layer.

Signature

object.SymbolLayer

  object
    DescriptionKey The object or objects this property applies to.

  SymbolLayer
    String; read-write
    Returns and sets the symbol layer.
SymbolManagerPath Property

Specifies the directory in which the symbol sets are located.

See Also | Example

Signature

object.SymbolManagerPath

object

PreferencesFiles The object or objects this property applies to.

SymbolManagerPath

String; read-write
The drive and path for the symbol sets.

Remarks

By default, this path is C:\Documents and Settings\All Users\Application Data\Autodesk\AutoCAD Land Desktop <version number>\<release number>\Data\Symbol Manager. You must exit AutoCAD Land Desktop and restart for this change to take effect.
SystemPath Property

Specifies the directory in which AutoCAD Land Desktop is installed.

See Also | Example

**Signature**

object.SystemPath

object

PreferencesFiles The object or objects this property applies to.

SystemPath

String; read-only
The drive and path where AutoCAD Land Desktop is installed.
TangentLabelIncrement Property (Civil Engineering Feature)

Returns the distance between tangent elevation labels.

See Also | Example

Signature

object.TangentLabelIncrement

object

ProfileBlock The object or objects this property applies to.

TangentLabelIncrement

Double; read-only
The Tangent Labeling Increment for the Profile.
TangentLength Property

The tangent length of the curve.

See Also | Example

**Signature: Overview**

- **AlignCurve**
- **ParcelCurve**

The tangent length of the alignment curve.

**Signature**

```object.TangentLength```

- **object**

  **AlignCurve** The object or objects this property applies to.

  **TangentLength**

  Double; read-only
  The tangent length of the curve.

The tangent length of the parcel curve.

**Signature**

```object.TangentLength```
object

ParcelCurve
The object or objects this property applies to.

TangentLength
Double; read-only
The tangent length of the curve.
**TempPath Property**

Specifies the directory in which the temporary files are stored.

See Also | Example

**Signature**

object.TempPath

object

preferencesFiles The object or objects this property applies to.

TempPath

String; read-write

The drive and path for the temporary files.

**Remarks**

By default, this path is c:\temp. You must exit AutoCAD Land Desktop and restart for this change to take effect.
TextAbove Property

The text above the object.

See Also | Example

Signature

object.TextAbove

object

AeccCurveText The object or objects this property applies to.

TextAbove

String; read-write
The text above the curve text object.
TextBelow Property

The text below the object.

See Also | Example

Signature

object.TextBelow

object

AeccCurveText The object or objects this property applies to.

TextBelow

String; read-write
The text below the curve text object.
**TextHeight Property**

The height of the text in the drawing.

See Also | Example

**Signature**

object.TextHeight

object

DatabasePreferences The object or objects this property applies to.

TextHeight

Double; read-write
The text height in inches or millimeters.

**Remarks**

The default value is 0.125 inches, or 3.175 millimeters, depending on the setting of the MeasurementUnit property.
TextOffsetAbove Property

The offset of the text above the object.

Signature

object.TextOffsetAbove

  object

  AeccCurveText The object or objects this property applies to.

  TextOffsetAbove

  Double; read-write
  The offset of the text above the curve text object.
TextOffsetBelow Property

The offset of the text below the object.

Signature

object.TextOffsetBelow

object

AeccCurveText The object or objects this property applies to.

TextOffsetBelow

Double; read-write
The offset of the text below the curve text object.
**TextSize Property**

The size of the text above and below the object.

**Signature**

object.TextSize

object

[AeccCurveText](#) The object or objects this property applies to.

**TextSize**

Double; read-write
The size of the text above and below the curve text object.
## ThetaExt Property

Returns the external theta for a compound spiral.

**Signature**

```
object.ThetaExt
```

- **object**
  - `AlignSpiral` The object or objects this property applies to.

- **ThetaExt**
  - Double; read-only
  - The external theta for a compound spiral.
Time Property

The lock creation date and time.

See Also | Example

Signature

object.Time

object

FileLock The object or objects this property applies to.

Time

String; read-only
The date and time of the file lock.
**TinPoints Property**

Returns the TinPoints collection.

**Signature**

`object.TinPoints`

object

*SurfaceOutputs* The object or objects this property applies to.

TinPoints

*TinPoints* collection; read-only

The *TinPoints* collection.
**TotalX Property**

The total X value for the spiral.

**Signature**

object.TotalX

- object
  - **AlignmentSpiral** The object or objects this property applies to.

- TotalX
  - Double; read-only
  - Total X value.

[See Also](#) | [Example](#)
TotalY Property

The total Y value for the spiral.

**Signature**

object.TotalY

  object  **AlignSpiral** The object or objects this property applies to.

TotalY

  Double; read-only
  Total Y value.

See Also | Example
Type Property

The type properties for Objects.

**Signature: Overview**

- AlignCurve, AlignEntity, AlignSpiral, AlignTangent
- Boundary
- BreakLine
- CrossSectionSurface (Civil Engineering Feature)
- EGProfile (Civil Engineering Feature)
- FGProfile (Civil Engineering Feature)
- ParcelCurve, ParcelLine
- StationEquation
- Surface
- Watershed

The type of the alignment entity.

**Signature**
object.type

object

AlignCurve, AlignEntity, AlignSpiral, AlignTangent
The object this property applies to.

Type

eAeccAlignEntityType enum; read-only
The type of the Alignment.

kCurve: Alignment Curve
kSpiral: Alignment Spiral
kTangent: Alignment Tangent

The type of the Boundary.

Signature

object.Type

object

Boundary
The object or objects this property applies to.

Type

eAeccBoundaryType enum; read-only
The type of the Boundary.

kBoundaryTypeShow: Boundary is visible
kBoundaryTypeHide: Boundary is hidden
kBboundaryTypeOuter: Boundary is the outer most boundary
**kBoundaryTypeUnInit:** Boundaries are not initialized

**Remarks**

A Boundary can not be changed to an outer (kBoundaryTypeOuter) boundary. If the Boundary is an outer, it is always visible.

---

The type of the BreakLine.

**Signature**

object>Type

object

BreakLine

The object or objects this property applies to.

**Type**

eAeccBreakLineType enum; read-only

The type of the BreakLine.

kStandard: Standard breakline

kProximity: Proximity breakline

kWallLeft: Left wall breakline

kWallRight: Right wall breakline

kNonDestructive: Nondestructive breakline

---

Returns the type of the cross section surface.

**Signature**
Object.Type

object

CrossSectionSurface
The object or objects this property applies to.

Type
eAeccCrossSectionSurfaceType enum; read-only
The type of cross section surface.

kExistingGround: 0

kTopSurface: 10

kDatumSurface: 20

kTemplateSurface: 30

kSubassemblySurface: 40

kMatchSurface: 50

----------------------------------

Returns the type of existing ground profile.

Signature

Object.Type

object

EGProfile
The object or objects this property applies to.

Type
eAeccEGProfileType enum; read-only
The VerticalOffset type.

kEgCenter: Existing ground center type
Returns the type of the Finished Ground Profile.

**Signature**

`Object.Type`

`object` **FGProfile**

The object or objects this property applies to.

**Type**

`eAeccFGProfileType` enum; read-only

Returns the type of the Finished Ground Profile.

- **kFgCenter**: Finished ground profile center type
- **kFgDitchLeft**: Finished ground ditch left type
- **kFgDitchRight**: Finished ground ditch right type
- **kFgLeft1**: Finished ground left 1 trans type
- **kFgLeft2**: Finished ground left 2 trans type
- **kFgLeft3**: Finished ground left 3 trans type
- **kFgLeft4**: Finished ground left 4 trans type
kFgLeft5: Finished ground left 5 trans type
kFgLeft6: Finished ground left 6 trans type
kFgLeft7: Finished ground left 7 trans type
kFgLeft8: Finished ground left 8 trans type
kFgNone: Finished ground none type
kFgRight1: Finished groung right 1 trans type
kFgRight2: Finished groung right 2 trans type
kFgRight3: Finished groung right 3 trans type
kFgRight4: Finished groung right 4 trans type
kFgRight5: Finished groung right 5 trans type
kFgRight6: Finished groung right 6 trans type
kFgRight7: Finished groung right 7 trans type
kFgRight8: Finished groung right 8 trans type

Returns the parcel entity type.

**Signature**

`object.Type`  
`object`  
   **ParcelCurve**, **ParcelLine**

The object or objects this property applies to.
The type of the parcel entity.

**Type**

eAeccParcelEntityType enum; read-only
The type of the parcel entity.

kParcelCurve: Parcel entity is a curve

kParcelLine: Parcel entity is a line

The type of station equation.

**Signature**

object.Type

object

**StationEquation**
The object or objects this property applies to.

**Type**
eAeccStationEquationType enum; read-write
The type of the Station Equation.

kIncreasing: Station Equation increases stationing

kDecreasing: Station Equation decreases stationing

The Surface type.

**Signature**

object.Type

object

**Surface**
The object or objects this property applies to.

**Type**

eAeccSurfaceType enum; read-only  
The type of the Surface.

- kUnknownSurface: Unknown surface type  
- kTin: Tin surface type  
- kCompositeVolume: Composite volume surface type  
- kGridVolume: Grid volume surface type

The type of the WaterShed.

**Signature**

object.Type  

object  

**WaterShed**  
The object or objects this property applies to.

**Type**

eAeccWaterShedType enum: read-only  
The type of the WaterShed.

- kBoundaryPoint: A watershed that includes a boundary point, the point where the channel of water drains off the surface  
- kBoundarySegment: A watershed that includes part of the boundary of a surface as its drain target
<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kLocalMin:</td>
<td>The watershed is a depression</td>
</tr>
<tr>
<td>kFlatArea:</td>
<td>A flat area where the water collects, passes over, or causes a channel to split. An ambiguous watershed.</td>
</tr>
<tr>
<td>kMultiDrain:</td>
<td>A watershed where the water can go to more than one drain target. An ambiguous watershed</td>
</tr>
<tr>
<td>kNonBoundary:</td>
<td></td>
</tr>
<tr>
<td>kUnknownSink:</td>
<td>A watershed formed by a notch and which drains to two or more drain targets</td>
</tr>
</tbody>
</table>

**Remarks**

Each watershed subarea that you delineate is categorized based on drain target.
UpperRight Property (Civil Engineering Feature)

Returns the Upper Right Coordinates of the ProfileBlock as XY.

**See Also** | **Example**

**Signature**

object.UpperRight

object

ProfileBlock The object or objects this property applies to.

UpperRight

Variant (two-element array of doubles); read-only
An array of the X and Y value, representing the upper right coordinates of a profile. This and the coordinates of the origin define the drawing limits of a profile.
UsedPointNumbers Property

Returns the list of used point numbers in the current project database.

See Also | Example

Signature

object.UsedPointNumbers()

object

CogoPoints The object or objects this property applies to.

UsedPointNumbers

String; read-only
Comma delimited string with all used points with groups separated by hyphens.

Remarks

Typical examples of the point string format are: "1,2,3,5,10" "1-100,1000-2000"
User Property

Gets the PreferencesUser object.

Signature

object.User

object

Preferences The object or objects this property applies to.

User

PreferencesUser object; read-only
Gets the PreferencesUser object.

Remarks

This object specifies the options maintained on a per-user basis.
Utility Property

Gets the Utility object.

See Also | Example

Signature

object.Utility

object

Document The object or objects this property applies to.

Utility

Utility object; read-only
The Utility object.
**VerticalScale Property**

Specifies the scale at which the vertical axis of the drawing is displayed, or the vertical scale for alignment profiles and cross sections.

See Also | Example

**Signature: Overview**

- **CrossSectionBlock** (Civil Engineering Feature)
- **DatabasePreferences**
- **ProfileBlock** (Civil Engineering Feature)

Specifies the vertical scale at which the alignment cross section is drawn.

**Signature**

```
object.VerticalScale
```

- `object` **CrossSectionBlock** The object or objects this property applies to.

```
VerticalScale
```

- Double; read-only
  The vertical scale at which the alignment cross section is drawn.
Specifies the scale at which the vertical axis of the drawing is displayed.

**Signature**

```csharp
object.VerticalScale
```

**Remarks**

The `VerticalScale` is compared against the `DatabaseScale` to calculate the vertical exaggeration in profiles and cross sections. It does not actually change the scale that is used when the drawing is plotted.

---

Specifies the vertical scale at which the alignment profiles are drawn.

**Signature**

```csharp
object.VerticalScale
```

---
Volume Property

The volume of the surface.

See Also | Example

**Signature**

object.Volume

object

**Surface** The object or objects this property applies to.

Volume

Double; read-only

The surface volume.

**Remarks**

This is an Extended Statistic.
**VolumeDisplayUnit Property**

The units used to display volumes.

---

**Signature**

```
object.VolumeDisplayUnit
```

- **object**
  - `DatabasePreferences` The object or objects this property applies to.

- **VolumeDisplayUnit**
  - `AecBuiltInUnit enum; read-write`
    - `aecUnitCubicInch`: Cubic inch unit
    - `aecUnitCubicFoot`: Cubic foot unit
    - `aecUnitCubicYard`: Cubic yard unit
    - `aecUnitCubicMillimeter`: Cubic millimeter unit
    - `aecUnitCubicCentimeter`: Cubic centimeter unit
    - `aecUnitCubicMeter`: Cubic meter unit

---

**See Also** | **Example**
VolumePrecision Property

The precision used to display volumes.

**Signature**

object.VolumePrecision

object

**DatabasePreferences**
The object or objects this property applies to.

VolumePrecision

Long; read-write
The display precision used for volumes.

**Remarks**

The precision settings are used only for labeling and listing values, not actual computations. AutoCAD Land Desktop commands always calculate all numbers up to the highest internal precision.
**VolumeSuffix Property**

The suffix displayed after an volume.

**Signature**

object.VolumeSuffix

object

[DatabasePreferences](#) The object or objects this property applies to.

VolumeSuffix

String; read-write

The suffix displayed after a volume.

**Remarks**

The default value is " Cu.Ft." or " m3", depending on the value of the [MeasurementUnit](#) property. The value must 255 characters or less in length.
Watersheds Property

Returns the WaterSheds collection

See Also | Example

Signature

object.WaterSheds

object  
  
  SurfaceOutputs  The object or objects this property applies to.

WaterSheds

WaterSheds  collection; read-only  
  The Watersheds collection.
A

**AeccApplication** An instance of the AutoCAD Land Desktop application.

**AeccContour** The AeccContour represents a contour entity in the document.

**AeccContourStyle** The AeccContourStyle represents a contour style entry in the dictionary.

**AeccCurveText** The AeccCurveText represents a curve text entity in the document.

**AeccPoint** The AeccPoint represents a cogo point entity in the document.

**AlignCurve** The AlignCurve object represents a curve entity for the alignment.

**AlignEntities** The AlignEntities collection represents all of the entities for an Alignment.

**AlignEntity** The AlignEntity object is the base class for AlignTangent, AlignCurve and AlignSpiral.

**Alignment** The Alignment object represents an alignment in the project database.

**Alignments** The Alignments collection represents all of the Alignment objects in the project database.

**AlignSpiral** The AlignSpiral object represents a spiral entity for the alignment.

**AlignTangent** The AlignTangent object represents a tangent entity for the alignment.
Boundaries The Boundaries collection represents all of the boundaries for the Surface.

Boundary The Boundary object represents a single boundary for the Surface.

BreakLine The BreakLine object represents a single breakline for the Surface.

BreakLines The BreakLines collection represents all of the breaklines for the Surface.

CogoPoint The CogoPoint represents a COGO point in the project database.

CogoPoints The CogoPoints collection represents all of the COGO points in the project database.

Contours The ContourItem object represents a single contour for the ContourItems collection.

ContourItems The ContourItems collection represents all of the input contours for the Surface.

CrossSection (Civil Engineering Feature) The CrossSection object represents a single section for an alignment.

CrossSectionBlock (Civil Engineering Feature) The CrossSectionBlock object represents an alignment cross section in the current drawing.

CrossSectionBlocks (Civil Engineering Feature) The CrossSectionBlocks collection represents all of the alignment cross sections that are in the current drawing.

CrossSectionPointCode (Civil Engineering Feature) The CrossSectionPointCode object represents a single point code for a cross...
section.

**CrossSectionPointCodes** (Civil Engineering Feature) The CrossSectionPointCodes collection represents all of the point codes for a cross section.

**CrossSections** (Civil Engineering Feature) The CrossSections collection represents all of the cross sections for an alignment.

**CrossSectionSurface** (Civil Engineering Feature) The CrossSectionSurface object represents a single surface section for a cross section.

**CrossSectionSurfaces** (Civil Engineering Feature) The CrossSectionSurfaces collection represents all of the surfaces for a cross section.

**DatabasePreferences** This object specifies the current AutoCAD Land Desktop drawing specific settings.

**DEMFile** The DEMFile object represents a single Digital Elevation Model file for the Surface.

**DEMFiles** The DEMFiles collection represents all of the Digital Elevation Model files for the Surface.

**DescriptionKey** The DescriptionKey object represents the description key for point.

**DescriptionKeyFile** The DescriptionKeyFile collection represents all of the description keys in a project description key file. Each key is contained in a particular description key file.

**DescriptionKeyFiles** The DescriptionKeyFiles collection represents all of the description keys for a project description key file.

**Document** An AutoCAD Land Desktop drawing.
**Documents** The collection of all AutoCAD Land Desktop drawings open in the current session.

**Drawing** An AutoCAD Land Desktop project-based drawing.

**Drawings** The collection of all AutoCAD Land Desktop drawings in a project.

**E**

**Edge** The Edge object represents a single output edge for the Surface.

**Edges** The Edges collection represents all of the output edges for the Surface.

**EGProfile** (Civil Engineering Feature) The EGProfile object represents the station / elevation information for a given surface.

**EGProfiles** (Civil Engineering Feature) The EGProfiles collection represents all of the existing ground vertical profiles.

**ElevationContour** The ElevationContour object represents a single contour for the ElevationContours collection.

**ElevationContours** The ElevationContours collection represents all of the calculated contours at a given elevation for the Surface.

**F**

**Face** The face object represents a single calculated face (triangle) for the Surface.

**Faces** The faces collection represents all of the calculated faces (triangles) for the Surface.

**FGProfile** (Civil Engineering Feature) The FGProfile object represents a finished ground vertical PVIs.

**FGProfiles** (Civil Engineering Feature) The FGProfiles collection represents all of the finished grade profiles in the project.
**FileLock** A file lock in the AutoCAD Land Desktop project.

**FileLocks** The collection of all file locks for a particular project.

---

**Parcel** The Parcel object represents a parcel in the current project database.

**ParcelCurve** The ParcelCurve object represents a curve entity for the parcel.

**ParcelEntities** The ParcelEntities collection represents all of the entities for a Parcel.

**ParcelEntity** The ParcelEntity collection is the base class for ParcelCurve and ParcelLine.

**ParcelLine** The ParcelLine object represents a line entity for the Parcel.

**Parcels** The Parcels collection represents all of the parcels that are in the current project.
**PointFile** The PointFile object represents a single point file for the Surface.

**PointFiles** The PointFiles collection represents all of the point files for the Surface.

**PointGroup** The PointGroup collection represents a Point Group in the project database.

**PointGroupName** The PointGroupName object represents a single PointGroupName for the PointGroupNames collection.

**PointGroupNames** The PointGroupNames collection represents all of the PointGroupNames for the Surface.

**PointGroups** The PointGroups collection represents all of the Point Groups in the project database.

**Preferences** This object specifies the current AutoCAD Land Desktop settings.

**PreferencesAlignment** This object specifies the settings for Horizontal Alignments.

**PreferencesCogo** This object specifies the settings for Points.

**PreferencesCrossSection** (Civil Engineering Feature) This object specifies the settings for Alignment Cross Sections.

**PreferencesFiles** This object specifies the paths used by the AutoCAD Land Desktop.

**PreferencesParcel** This object specifies the settings for Parcels.

**PreferencesProfile** (Civil Engineering Feature) This object specifies the settings for Alignment Vertical Profiles.

**PreferencesProject** This object specifies the project settings.

**PreferencesSurface** This object specifies the settings for Surfaces.
PreferencesUser This object specifies the options maintained on a per-user basis.

ProfileBlock (Civil Engineering Feature) The ProfileBlock object represents a profile in the current drawing.

ProfileBlocks (Civil Engineering Feature) The ProfileBlocks collection represents all of the profiles that are in the current drawing.

Project An AutoCAD Land Desktop project.

Projects The collection of all AutoCAD Land Desktop projects on the network.

Prototype An AutoCAD Land Desktop project prototype.

Prototypes The collection of all AutoCAD Land Desktop project prototypes.

PVI (Civil Engineering Feature) The PVI object represents a Point of Vertical Intersection for a Finished Ground Profile.

PVIs (Civil Engineering Feature) The PVIs collection represents all of the Point of Vertical Intersection objects in the project.

Q

R

S

StationEquation The StationEquation object represents a station equation for the alignment.

StationEquations The StationEquations collection represents all of the station equations for the Alignment.

Superelevation (Civil Engineering Feature) The Superelevation object represents a single superelevation for an alignment.

Superelevations (Civil Engineering Feature) The Superelevations
collection represents all of the superelevations for a cross section.

**Surface** The Surface object represents a Surface in the project database.

**SurfaceInputs** The SurfaceInputs object represents all inputs for a Surface.

**SurfaceOutputs** The SurfaceOutputs object represents all outputs for a Surface.

**Surfaces** The Surfaces collection represents all of the Surface objects in the project database.

T

**TinPoint** The TinPoint represents a calculated point for the Surface.

**TinPoints** The TinPoints represents all of the calculated points for the Surface.

U

**Utility** A series of methods provided for utility purposes.

V

W

**WaterShed** The WaterSheds collection represents all of the WaterShed items for the Surface.

**WaterSheds** The WaterShed object represents a single watershed item for the WaterSheds collection.

X

Y

Z
An instance of the AutoCAD Land Desktop application.

**VBA object name:** AeccApplication

**Create using:**

For VB:

GetObject("Acad.Application") or
reateObject("Acad.Application") then

etInterfaceObject("Aecc.Application")

For AutoCAD VBA:
not applicable. The application is always available.

**Access via:**  
Application Property

The properties associated with the AeccApplication object reflect the properties of the main application window. The methods control the loading or listing of the currently loaded external applications and interface objects.

The active document (AutoCAD Land Desktop drawing) can be accessed using the ActiveDocument property. The active project (AutoCAD Land
Desktop project) can be accessed using the `ActiveProject` property.

See the [Application object](#) in the AutoCAD ActiveX and VBA Reference for information regarding additional Methods, Properties and Events provided through this object.

Properties

- **ActiveDocument**
- **ActiveProject**

Methods

- **Application**

Events

- None

Preferences

Projects

Prototypes
The AeccContour represents a contour entity in the document.

**VBA object name**: AeccContour

**Create using**: `AcadModelSpace.AddCustomObject("AeccDbContour")`

**Access via**: `AcadModelSpace.Item`

In addition to the properties listed below, the AeccContour object supports all of the members of the **AcadEntity** object.

<table>
<thead>
<tr>
<th>Properties</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AddLabelAt</td>
<td></td>
</tr>
<tr>
<td>RemoveAllLabels</td>
<td></td>
</tr>
<tr>
<td>RemoveLabelAt</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Events</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ContourStyleName</td>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Coordinates</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Elevation</td>
<td></td>
</tr>
</tbody>
</table>
Elevation

Label Points
The AeccContourStyle represents a contour style entry in the dictionary.

**VBA object name:** AeccContourStyle

**Create using:** N/A

**Access via:** ThisDrawing.Dictionaries("AECC_CONTOUR_STYLES").Item

In addition to the properties listed below, the AeccContourStyle object supports all of the members of the AcadObject object.

The contour styles dictionary is not added to the drawing until a contour object is created, or when the contour style manager is used. The "Standard" contour style is created by default.

**Properties**

- Application
- Name

**Methods**

- Application

**Events**

- None
The AeccCurveText represents a curve text entity in the document.

**VBA object name:** AeccCurveText

**Create using:** `AcadModelSpace.AddCustomObject("AecDbCurveText")`

**Access via:** `AcadModelSpace.Item`

In addition to the properties listed below, the AeccCurveText object supports all of the members of the `AcadEntity` object.

<table>
<thead>
<tr>
<th>Properties</th>
<th></th>
<th>Methods</th>
<th></th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td></td>
<td>SetReferenceCurve</td>
<td>TextBelow</td>
<td>None</td>
</tr>
<tr>
<td>TextAbove</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TextOffsetAbove</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TextBelow</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TextOffsetBelow</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### AeccPoint Object

The AeccPoint represents a cogo point entity in the document.

**VBA object name:** AeccPoint

**Create using:**

```vba
AcadModelSpace.AddCustomObject("AeccDbPoint;")
```

**Access via:**

`AcadModelSpace.Item`

In order to have AeccPoints visible in the drawing using `AddCustomObject()` and setting properties, the Number property is required to be set.

In addition to the properties listed below, the AeccPoint object supports all of the members of the AcadEntity object.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Application</strong></td>
<td><strong>Easting</strong></td>
<td><strong>Events</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

AlignCurve Object

The AlignCurve object represents a curve entity for the alignment.

VBA object name: AeccAlignCurve

Create using: Alignment.AddCurve

Access via: AlignEntities.Item

Properties

Application
CCWFlag
CenterEasting
CenterNorthing
ChordDirection
ChordLength
Delta
EndDirection
EndEastings
EndingStation
EndNorthing
ExternalSecant
Length
MidOrdinate
ObjectId
PiEastings
PiNorthing
Radius
StartDirection
StartEastings
StartingStation
StartNorthing
TangentLength
Type
The AlignEntities collection represents all of the entities for an Alignment.

**VBA object name:** AeccAlignEntities

**Create using:** N/A

**Access via:** Alignment.AlignEntities

This collection returns AlignTangent, AlignCurve and AlignSpiral objects. When you access a member of the collection using the `Item` method or the For Each statement, assign the result to a VARIANT. You can then use the `TypeOf` function to determine which object type (AlignTangent, AlignCurve or AlignSpiral) to assign it to.

### Properties
- Application
- Count

### Methods
- Item

### Events
- Modified
The AlignEntity object is the base class for AlignTangent, AlignCurve and AlignSpiral.

**VBA object name:** AeccAlignEntity

**Create using:** N/A

**Access via:** AlignEntities.Item

All alignment entities are derived from AlignEntity - they all share the methods, properties and events listed below. You can use this base class as the object type returned from the method Item of the AlignEntities collection. If you need access to the complete details of an alignment entity, use the Type property (or TypeOf function in VB) to determine whether to assign the returned object to an AlignTangent, AlignCurve or AlignSpiral.

### Properties

- **Application**
<table>
<thead>
<tr>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>Modified</td>
</tr>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

- **EndEastng**
- **EndingStation**
- **EndNorthing**
- **ObjectID**
- **StartEasting**
- **StartingStation**
- **StartNorthing**
- **Type**
The Alignment object represents an Alignment in the project database.

**VBA object name:** AeccAlignment

**Create using:** Alignments.Add

**Access via:** Alignments.Item

The Alignment object exposes alignment geometry through the **AlignEntities** collection. Each AlignEntity (an **AlignTangent**, **AlignCurve** or **AlignSpiral**) exposes geometric properties, as well as the **ObjectID** of the corresponding entity in the current drawing (if any).

Valid alignment ObjectIDs are defined during an **Import**. The ObjectIDs are valid until the alignment is saved. Alignments are automatically saved each time an alignment property is modified when **AutoSave** is set to true, or when an alignment **Save** is executed.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>AlignEntities</strong></td>
</tr>
</tbody>
</table>
AddCurve
AddSpiral
AddTangent
ExternalStaToInternal
GetStaStrWithEquations
GetStaWithEquations
Import
LineIntersection
PerpIntersection
PointLocation
RemoveAll
Save
StationOffset

Application

CrossSections (Civil Engineering Feature)

Description

EGProfiles (Civil Engineering Feature)

EndingStation

EGProfiles (Civil Engineering Feature)

LockType

Name

Number

StartingStation

StationEquations

Superelevations (Civil Engineering Feature)
The Alignments collection represents all of the Alignment objects in the project database.

**VBA object name:** AeccAlignments

**Create using:** N/A

**Access via:** Project.Alignments

All Alignments except for the current Alignment are loaded with read only permissions. The current alignment can have read or read-write permissions. If no other user has control of the current alignment, the read-write permissions will be given. If another user has read-write permission, then only read permission will be given. As you switch the current Alignment, permissions will be updated. Also, by setting the current Alignment, you are also setting the current Alignment for the AutoCAD Land Desktop menus.

**Methods**

- **Add**

**Properties**
<table>
<thead>
<tr>
<th>AlignmentFromObjectID</th>
<th>Application</th>
<th>Events</th>
<th>CurrentAlignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete</td>
<td>AutoSave</td>
<td>Modified</td>
<td></td>
</tr>
<tr>
<td>DoubleToStaFormat</td>
<td>Count</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The AlignSpiral object represents a spiral entity for the alignment.

**VBA object name:** AeccAlignSpiral

**Create using:** Alignment.AddSpiral

**Access via:** AlignEntities.Item

### Properties

<table>
<thead>
<tr>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
</tr>
<tr>
<td>AD1</td>
</tr>
<tr>
<td>AD2</td>
</tr>
<tr>
<td>Application</td>
</tr>
<tr>
<td>BeginCondition</td>
</tr>
<tr>
<td>Delta</td>
</tr>
<tr>
<td>Methods</td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>EndEasting</td>
<td>StartingEnd</td>
<td></td>
</tr>
<tr>
<td>EndingStation</td>
<td>EndNorthing</td>
<td></td>
</tr>
<tr>
<td>ExtEasting</td>
<td>EndNorthing</td>
<td></td>
</tr>
<tr>
<td>ExtNorthing</td>
<td>ExtEasting</td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>ExtNorthing</td>
<td></td>
</tr>
<tr>
<td>Length</td>
<td>K</td>
<td></td>
</tr>
<tr>
<td>LExt</td>
<td>K</td>
<td></td>
</tr>
<tr>
<td>LOffset</td>
<td>Length</td>
<td></td>
</tr>
<tr>
<td>RadialDistance</td>
<td>LExt</td>
<td></td>
</tr>
<tr>
<td>ShortTangent</td>
<td>LOffset</td>
<td></td>
</tr>
<tr>
<td>SpiEasting</td>
<td>ObjectId</td>
<td></td>
</tr>
<tr>
<td>SpilTangent</td>
<td>Modified</td>
<td></td>
</tr>
<tr>
<td>SpiNorthing</td>
<td>SpiEasting</td>
<td></td>
</tr>
<tr>
<td>SpiralType1</td>
<td>SpilTangent</td>
<td></td>
</tr>
<tr>
<td>SpiralType2</td>
<td>SpiNorthing</td>
<td></td>
</tr>
<tr>
<td>StartDirection</td>
<td>SpiralType1</td>
<td></td>
</tr>
<tr>
<td>StartEasting</td>
<td>SpiralType2</td>
<td></td>
</tr>
<tr>
<td>StartingStation</td>
<td>StartNorthing</td>
<td>ThetaExt</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------</td>
<td>---------</td>
</tr>
</tbody>
</table>
The AlignTangent object represents a tangent entity for the alignment.

**VBA object name:** AeccAlignTangent

**Create using:** Alignment.AddTangent

**Access via:** AlignEntities.Item

<table>
<thead>
<tr>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
</tr>
<tr>
<td>Direction</td>
</tr>
<tr>
<td>EndEastings</td>
</tr>
<tr>
<td>EndingStation</td>
</tr>
<tr>
<td>EndNorthing</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Length</td>
</tr>
<tr>
<td></td>
<td>Modified</td>
</tr>
</tbody>
</table>
ObjectID
StartEasting
StartingStation
StartNorthing
Type
The Boundaries collection represents all of the boundaries for the current Surface.

**VBA object name:** AeccBoundaries

**Create using:** N/A

**Access via:** SurfaceInputs.Boundaries

There are three types of boundaries: outer, hidden, and visible. If an outer boundary exists for the Surface, it must be the first boundary and it is visible. Also, a boundary can not be changed from a hidden / visible boundary to an outer boundary.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Application</td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td>Count</td>
<td>Modified</td>
</tr>
<tr>
<td>Item</td>
<td>VerticalScale</td>
<td></td>
</tr>
</tbody>
</table>
The Boundary object represents a single boundary for the Surface.

**VBA object name:** AeccBoundary

**Create using:** Boundaries.Add

**Access via:** Boundaries.Item

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Description</td>
<td>Modified</td>
</tr>
<tr>
<td>Coordinates</td>
<td>Id</td>
<td></td>
</tr>
<tr>
<td>Id</td>
<td>IsBreakLine</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The BreakLine object represents a single breakline for the Surface.

**VBA object name**: AeccBreakLines

**Create using**: BreakLines.Add

**Access via**: BreakLines.Item

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Coordinates</td>
<td>Description</td>
</tr>
<tr>
<td>Id</td>
<td>Modified</td>
<td>Type</td>
</tr>
</tbody>
</table>
The BreakLines collection represents all of the breaklines for the Surface.

**VBA object name:** AeccBreakLines

**Create using:** N/A

**Access via:** SurfaceInputs.BreakLines

**Methods**
- Add
- Delete

**Properties**
- Application
- Count
- Item

**Events**
- Modified
The CogoPoint represents a COGO point in the project database.

**VBA object name:** AeccCogoPoint

**Create using:** CogoPoints>Add

**Access via:** CogoPoints>Item

**Properties**

- Application
- Coordinates
- Easting
- Elevation
- FullDescription
- GridEasting
<table>
<thead>
<tr>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>GridNorthing</td>
<td>Modified</td>
</tr>
<tr>
<td>GroupName</td>
<td></td>
</tr>
<tr>
<td>Save</td>
<td></td>
</tr>
<tr>
<td>Latitude</td>
<td></td>
</tr>
<tr>
<td>LockType</td>
<td></td>
</tr>
<tr>
<td>Longitude</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>Northing</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td></td>
</tr>
<tr>
<td>OverrideDescription</td>
<td></td>
</tr>
<tr>
<td>OverrideElevation</td>
<td></td>
</tr>
<tr>
<td>OverrideName</td>
<td></td>
</tr>
<tr>
<td>RawDescription</td>
<td></td>
</tr>
</tbody>
</table>
The CogoPoints collection represents all of the COGO points in the project database.

**VBA object name:** AeccCogoPoints

**Create using:** N/A

**Access via:** Project.CogoPoints

The CogoPoints collection represents all Cogo points for the current project database. There exists only one CogoPoints collection for at a time.

**Methods**
- Add
- ArrayToPointString
- Delete

**Properties**
- Application
- AutoSave
- Count
<table>
<thead>
<tr>
<th>Item</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>LockPoints</td>
<td>IsNameSupported, Modified</td>
</tr>
<tr>
<td>PointByNumber</td>
<td>LockedPointNumbers</td>
</tr>
<tr>
<td>PointNumberFromObjID</td>
<td>NextPointNumber</td>
</tr>
<tr>
<td>PointStringToArray</td>
<td>PointNameSize</td>
</tr>
<tr>
<td>UnlockPoints</td>
<td>UsedPointNumbers</td>
</tr>
</tbody>
</table>
The ContourItem object represents a single contour for the ContourItems collection.

**VBA object name:** AeccContourItem

**Create using:** ContourItems.Add

**Access via:** ContourItems.Item

**Properties**
- Application
- Coordinates
- Id
- Modified

**Methods**
- None
The ContourItems collection represents all of the input contours for the Surface.

**VBA object name:** AeccContourItems

**Create using:** N/A

**Access via:** SurfaceInputs.ContourItems

**Methods**
- Add
- Delete
- Count

**Properties**
- Application

**Events**
- Modified
- Modified
The CrossSection object represents a single section for an alignment.

**VBA object name:** AeccCrossSection

**Create using:** N/A

**Access via:** CrossSections.Item

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>CrossSectionSurfaces</td>
<td>Modified</td>
</tr>
<tr>
<td>CrossSectionPointCodes</td>
<td>Import</td>
<td>MaxElevation</td>
</tr>
<tr>
<td>SectionVolume</td>
<td>MaxOffset</td>
<td></td>
</tr>
</tbody>
</table>
MinElevation

MinOffset

Station
The CrossSectionBlock object represents an alignment cross section in the current drawing.

**VBA object name:** AeccCrossSectionBlock

**Create using:** N/A

**Access via:** CrossSectionBlocks.Item

### Properties
- Application
- CenterlineOffset
- Coordinates
- DatumElevation
- EGPPrecision

### Methods
<table>
<thead>
<tr>
<th>GetLayerName</th>
<th>FGPrecision</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>OffsetElevationToXY</td>
<td>Height</td>
<td>Modified</td>
</tr>
<tr>
<td>XyToOffsetElevation</td>
<td>LeftWidth</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>RightWidth</td>
<td></td>
</tr>
<tr>
<td>Station</td>
<td>VerticalScale</td>
<td></td>
</tr>
</tbody>
</table>
The CrossSectionBlocks collection represents all of the alignment cross sections that are in the current drawing.

**VBA object name:** AeccCrossSectionBlocks

**Create using:** N/A

**Access via:** Document.CrossSectionBlocks

**Properties**
- Application
- Count

**Methods**
- Item

**Events**
- Modified
The CrossSectionPointCode object represents a single point code for a cross section.

**VBA object name:** AeccCrossSectionPointCode

**Create using:** N/A

**Access via:** CrossSectionPointCodes.Item

**Properties**

- Application

**Methods**

None

**Events**

- Description
- Modified

**None**

- Elevation
- Offset
The CrossSectionPointCodes collection represents all of the point codes for a cross section.

**VBA object name:** AeccCrossSectionPointCodes

**Create using:** N/A

**Access via:** CrossSection.CrossSectionPointCodes

**Properties**
- Application

**Methods**
- Item

**Events**
- Modified
The CrossSections collection represents all of the cross sections for an alignment.

**VBA object name:** AeccCrossSections

**Create using:** N/A

**Access via:** Alignments.CrossSections

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>Application</td>
<td>Modified</td>
</tr>
<tr>
<td>PointCodeDescription</td>
<td>Count</td>
<td></td>
</tr>
<tr>
<td>SectionByStation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The CrossSectionSurface object represents a single surface section for a cross section.

**VBA object name:** AeccCrossSectionSurface

**Create using:** N/A

**Access via:** CrossSectionSurfaces.Item

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Name</td>
<td>Modified</td>
</tr>
<tr>
<td>OffsetElevations</td>
<td>Type</td>
<td></td>
</tr>
</tbody>
</table>
The `CrossSectionSurfaces` collection represents all of the surfaces for a cross section.

**VBA object name:** `AeccCrossSectionSurfaces`

**Create using:** N/A

**Access via:** `CrossSection.CrossSectionSurfaces`

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Item</td>
<td>Count</td>
</tr>
</tbody>
</table>
This object specifies the current AutoCAD Land Desktop drawing specific settings.

**VBA object name:** AeccDatabasePreferences

**Create using:** N/A

**Access via:** Document.Preferences

Database preferences represent all the options that reside in a drawing. These include settings in the AdCADD block, dictionary and the AecBaseSetup object.

The DatabasePreferences object can be referenced from the Preferences property on the Document object.

See the DatabasePreferences object in the AutoCAD ActiveX and VBA Reference for information regarding additional Methods, Properties and Events provided through this object.
Methods

ConvertToCurrentAreaDisplay
ConvertToCurrentVolumeDisplay
LoadSetupProfile
SaveAsDefault

Properties

AngularAzimuth
AngularDisplayFormat
AngularPrecision
Application
AreaDisplayUnit
AreaPrecision
AreaSuffix
BasePoint
BasePointNE
BorderBlockFilename
BorderBottomMargin
BorderLeftMargin
BorderLineWidth
BorderRightMargin
BorderStyle
BorderTopMargin
CoordinatePrecision
CoordinateZone
DatabaseScale

Events
None
SaveSetupProfile

ElevationPrecision

FacetDeviation

LayerFile

LayerStandard

LinearDisplayFormat

LinearPrecision

LinearUnit

MeasurementUnit

NorthRotation

ProjectName

ScaleOnInsert

SheetHeight

SheetWidth

TextHeight

VerticalScale

VolumeDisplayUnit

VolumePrecision

VolumeSuffix
The DEMFile object represents a single Digital Elevation Model file for the Surface.

**VBA object name:** AeccDEMFile

**Create using:** DEMFiles.Add

**Access via:** DEMFiles.Item

### Properties
- Application
- CoordinateZone
- Modified
- Name

### Methods

### Events
The DEMFiles collection represents all of the Digital Elevation Model files for the Surface.

**VBA object name:** AeccDEMFiles

**Create using:** N/A

**Access via:** SurfaceInputs.DEMFiles.

**Methods**
- Add
- Delete
- Item

**Properties**
- Application
- Count

**Events**
- Modified
- Modified

**Functions**
- Count
- Count
- Count

**Defined Constants**
- Count
- Count
- Count
DescriptionKey Object

The DescriptionKey object represents the description key for point.

**VBA object name:** AeccDescriptionKey

**Create using:** DescriptionKeyFile.Add

**Access via:** DescriptionKeyFile.Item

**Properties**

- **Application**
- **Code**
- **DescriptionFormat**
- **DescriptionLayer**
- **RotateByDescriptionParam**
- **RotateByFixedFactor**
<table>
<thead>
<tr>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>RotateClockwise</td>
<td>Modified</td>
</tr>
<tr>
<td>RotateDescriptionParam</td>
<td></td>
</tr>
<tr>
<td>RotateFixedFactor</td>
<td></td>
</tr>
<tr>
<td>ScaleByDescriptionParam</td>
<td></td>
</tr>
<tr>
<td>ScaleByDrawingScale</td>
<td></td>
</tr>
<tr>
<td>ScaleByFixedFactor</td>
<td></td>
</tr>
<tr>
<td>ScaleDescriptionParam</td>
<td></td>
</tr>
<tr>
<td>ScaleFixedFactor</td>
<td></td>
</tr>
<tr>
<td>ScaleInXY</td>
<td></td>
</tr>
<tr>
<td>ScaleInZ</td>
<td></td>
</tr>
<tr>
<td>SymbolBlock</td>
<td></td>
</tr>
<tr>
<td>SymbolLayer</td>
<td></td>
</tr>
</tbody>
</table>
The DescriptionKeyFile collection represents all of the description keys in a project description key file.

**VBA object name:** AeccDescriptionKeyFile

**Create using:** DescriptionKeyFiles.Add

**Access via:** DescriptionKeyFiles.Item

**Properties**
- Application
- Count
- Delete
- FullName
- Modified
- Name
- Path
- Count
- Delete
- FullName
- Modified
- Name
- Path
The DescriptionKeyFiles collection represents all of the description keys for a project description key file.

**VBA object name:** AeccDescriptionKeyFiles

**Create using:** N/A

**Access via:** Project.DescriptionKeyFiles
An AutoCAD Land Desktop drawing.

VBA object name: AeccDocument

Create using: Documents.NewProjectBased

Access via: Documents.Item

Because AutoCAD Land Desktop works with only one document at a time, and that document represents one database, the active document can be thought of as the current database as well as the current document.

The NewProjectBased and OpenProjectBased methods handle project-based drawings specific to AutoCAD Land Desktop and override the default AutoCAD functionality.

The Preferences property provides access to drawing settings.

See the Document object in the AutoCAD ActiveX and VBA Reference for information regarding additional Methods, Properties and Events
provided through this object.

<table>
<thead>
<tr>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
</tr>
<tr>
<td>CrossSectionBlocks  (Civil Engineering Feature)</td>
</tr>
<tr>
<td>MaskBlockStyles</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>NewProjectBased</td>
</tr>
<tr>
<td>MassGroups</td>
</tr>
<tr>
<td>OpenProjectBased</td>
</tr>
<tr>
<td>MVBblockStyles</td>
</tr>
<tr>
<td>Preferences</td>
</tr>
<tr>
<td>ProfileBlocks        (Civil Engineering Feature)</td>
</tr>
<tr>
<td>ProfileStyles</td>
</tr>
<tr>
<td>Utility</td>
</tr>
</tbody>
</table>
The collection of all AutoCAD Land Desktop drawings open in the current session.

**VBA object name:** AeccDocuments

**Create using:** N/A

**Access via:** Application.Documents

The documents collection will always have a single item – the currently opened drawing. To open a different drawing, use the **OpenProjectBased** method from the **Document** object.

See the **Documents** object in the AutoCAD ActiveX and VBA Reference for information regarding additional Methods, Properties and Events provided through this object.
Count
An AutoCAD Land Desktop project-based drawing.

VBA object name: AeccDrawing

Create using: N/A

Access via: Drawings.Item

The Drawing object represents an AutoCAD drawing that is associated with a AutoCAD Land Desktop project.

Properties

<table>
<thead>
<tr>
<th>Application</th>
<th>None</th>
</tr>
</thead>
</table>

Methods

<table>
<thead>
<tr>
<th>None</th>
<th>None</th>
</tr>
</thead>
</table>

Events

<table>
<thead>
<tr>
<th>None</th>
</tr>
</thead>
</table>
The collection of all AutoCAD Land Desktop drawings in a project.

**VBA object name:**  AeccDrawings

**Create using:**  N/A

**Access via:**  Project.Drawings

The collection of drawings at the specified drawing path. Unlike the **Documents** collection, items in the Drawings collection are not necessarily loaded into the application. Use the Drawings collection to enumerate the drawings in the project, select a Project-based drawing to open, or create a new Project-based drawing from a template.

**Properties**

- **Application**

**Methods**

- **Item**
- **Count**

**Events**

- **Modified**

**Open**  **Path**
ShowSubfolders
The Edge object represents a single output edge for the Surface.

**VBA object name:** AeccEdge

**Create using:** N/A

**Access via:** Edges.Item

**Properties**
- Application
- Coordinates
- Modified

**Methods**
- None

**Events**
- None
The Edges collection represents all of the output edges for the Surface.

**VBA object name:** AeccEdges

**Create using:** N/A

**Access via:** SurfaceOutputs.Edges

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Count</td>
<td>Modified</td>
</tr>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The EGProfile object represents the station / elevation information for a given surface.

**VBA object name:** AeccEGProfile

**Create using:** PGProfiles.Add

**Access via:** EGProfiles.Item

**Properties**
- Application
- StationElevations
- SurfaceName
- Type

**Methods**
- ElevationAt
- Import
- InstantGrade

**Events**
- Modified
EGProfiles Collection (Civil Engineering Feature)

The EGProfiles collection represents all of the existing ground profiles.

**VBA object name:** AeccEGProfiles

**Create using:** N/A

**Access via:** Alignment.EGProfiles

**Methods**
- Add
- Delete
- Item
- ProfileByType

**Properties**
- Application
- Count

**Events**
- Modified
The ElevationContour object represents a single contour for the ElevationContours collection.

**VBA object name:** AeccElevationContour

**Create using:** N/A

**Access via:** ElevationContours.Item

Properties:
- Application
- Modified
- Coordinates

Methods:
- None
The ElevationContours collection represents all of the calculated contours at a given elevation for the Surface.

**VBA object name:** AeccElevationContours

**Create using:** N/A

**Access via:** SurfaceOutputs.ElevationContours

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>Count</td>
<td>Modified</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Elevation</td>
</tr>
</tbody>
</table>
The Face object represents a single calculated face (triangle) for the Surface.

**VBA object name:** AeccFace

**Create using:** N/A

**Access via:** Faces.Item

The Face object represents a single face (triangle) for the current TIN Surface. The Face is a three-dimensional (planar) surface triangle.

**Properties**

- Application
- Area2D
- Area3D

**Methods**

- None

**Events**

- Modified

**Coordinates**
The Faces collection represents all of the calculated faces (triangles) for the Surface.

**VBA object name:** AeccFaces

**Create using:** N/A

**Access via:** SurfaceOutputs.Faces

### Methods
- `FindAllFaces`
- `FindFace`
- `FindPath`
- `Item`

### Properties
- `Application`
- `Count`
- `SearchType`
- `Modified`

### Events
- `Modified`
The FGProfile object represents a finished ground vertical PVIs.

**VBA object name:** AeccFGProfile

**Create using:** FGProfiles.Add

**Access via:** FGProfiles.Item

**Methods**
- ElevationAt
- InstantGrade

**Properties**
- Application
- PVIs
- Type

**Events**
- Modified
The FGProfiles collection represents all of the finished grade profiles in the project.

**VBA object name:** AeccFGProfiles

**Create using:** N/A

**Access via:** Alignment.FGProfiles

**Methods**
- Add
- Delete
- Item
- ProfileByType

**Properties**
- Application
- Count
- Modified
- Name
FileLock Object

A file lock in the AutoCAD Land Desktop project.

**VBA object name:** AeccFileLock

**Create using:** N/A

**Access via:** FileLocks.Item

Use this object to access properties of a particular file lock.

**Properties**
- Application
- File
- LockType
- Owner

**Methods**
- Label
- None

**Events**
- Modified
Time
The collection of all file locks for a particular project.

**VBA object name:** AeccFileLocks

**Create using:** N/A

**Access via:** Project.FileLocks

The collection of all file locks for a particular project. Use the FileLocks collection to view which files in a project are locked, and who currently owns the locks. The lock owner is the person who opens the file first, locking the project files so that other people cannot make changes to them.

The delete methods are hidden since if used carelessly they can cause data corruption or loss of data. However, advanced users may require this functionality in the case of a power failure or system error.
The Parcel object represents a parcel in the current project database.

**VBA object name:** AeccParcel

**Create using:** Parcels.Add

**Access via:** Parcels.Item

### Properties
- Application
- Area
- CentroidEast
- CentroidNorthing
- Name
- Number

### Methods
- AddCurve
- AddLine
- Import

### Events
- Modified
ParcelEntities
Perimeter
Precision
The ParcelCurve object represents a curve entity for the parcel.

**VBA object name:** AeccParcelCurve

**Create using:** Parcel.AddCurve

**Access via:** ParcelEntities.Item

**Properties**

- Application
- CCWFlag
- CenterEasting
- CenterNorthing
- ChordLength
- Course
<table>
<thead>
<tr>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Delta</td>
</tr>
<tr>
<td></td>
<td>Modified</td>
</tr>
</tbody>
</table>

- CourseIn
- CourseOut
- Delta
- Modified

- EndEasting
- EndNorthing
- Length
- Radius
- StartEasting
- StartNorthing
- TangentLength
- Type
The ParcelEntities collection represents all of the entities for a Parcel.

**VBA object name**: AeccParcelEntities

**Create using**: N/A

**Access via**: Parcel.ParcelEntities

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Item</td>
<td>Modified</td>
</tr>
<tr>
<td>Count</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The ParcelEntity collection is the base class for ParcelCurve and ParcelLine.

**VBA object name:** AeccParcelEntity

**Create using:** N/A

**Access via:** ParcelEntities.Item

All parcel entities are derived from a ParcelEntity - they all share the methods, properties and events listed below. You can use this base class as the object type returned from the method Item of the ParcelEntities collection. If you need access to the complete details of a parcel entity, use the Type property (or TypeOf function in VB) to determine whether to assign the returned object to an ParcelCurve or ParcelLine.

<table>
<thead>
<tr>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
</tr>
<tr>
<td>Course</td>
</tr>
</tbody>
</table>
The ParcelLine object represents a line entity for the Parcel.

**VBA object name:** AeccParcelLine

**Create using:** Parcel.AddLine

**Access via:** ParcelEntities.Item

**Properties**
- Application
- Course
- EndEastings
- EndNorthing
- Length
- StartEastings

**Methods**
- None

**Events**
- Modified
StartNorthing

Type
The Parcels collection represents all of the parcels that are in the current project.

**VBA object name**: AeccParcels

**Create using**: N/A

**Access via**: Project.Parcels

**Methods**
- Add
- Delete
- Item
- Rename

**Properties**
- Application
- Count

**Events**
- Modified
- Modified
The PointFile object represents a single point file for the Surface.

**VBA object name:** AeccPointFile

**Create using:** PointFiles.Add

**Access via:** PointFiles.Item

**Properties**
- Application
- Format
- Modified
- Name

**Methods**
- None
The PointFiles collection represents all of the point files for the Surface.

**VBA object name:** AeccPointFiles

**Create using:** N/A

**Access via:** SurfaceInputs.PointFiles

**Methods:**
- **Add**
- **Delete**

**Properties:**
- Application

**Events:**
- Modified
- Count
- Item

**Methods:**
- **Add**
- **Delete**

**Properties:**
- Application

**Events:**
- Modified
- Count
- Item
The PointGroup collection represents a Point Group in the project database.

**VBA object name:** AeccPointGroup

**Create using:** PointGroups.Add

**Access via:** PointGroups.Item

**Properties**

- Application
- Description
- DescriptionOverride
- DescriptionXDRRef
- Elevation
- ElevationOverride
<table>
<thead>
<tr>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClearOverrides</td>
<td>Modified</td>
</tr>
<tr>
<td>Save</td>
<td></td>
</tr>
<tr>
<td>ElevationXDRef</td>
<td></td>
</tr>
<tr>
<td>GroupName</td>
<td></td>
</tr>
<tr>
<td>LabelStyle</td>
<td></td>
</tr>
<tr>
<td>LabelStyleOverride</td>
<td></td>
</tr>
<tr>
<td>LabelStyleXDRef</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>NameOverride</td>
<td></td>
</tr>
<tr>
<td>NameXDRef</td>
<td></td>
</tr>
<tr>
<td>PointList</td>
<td></td>
</tr>
</tbody>
</table>
The PointGroupName object represents a single PointGroupName for the PointGroupNames collection.

**VBA object name:** AeccPointGroupName

**Create using:** PointGroupNames.Add

**Access via:** PointGroupNames.Item

By using PointGroupNames, you can add Cogo Point Groups to the Surface definition. A Cogo Point Group will contain a range of Cogo Points. By using these objects, a Surface can be built using ranges of Cogo Points.

No validation exists between the PointGroupName and a Cogo Point Group. When a PointGroupName is added, AutoCAD Land Desktop does not check to see if the PointGroupName.Name actually exists as a Cogo Point Group.
<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Application</strong></td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
The PointGroupNames collection represents all of the PointGroupNames for the current Surface.

**VBA object name:** AeccPointGroupNames

**Create using:** N/A

**Access via:** SurfaceInputs.PointGroupNames

By using PointGroupNames, you can add Cogo Point Groups to the Surface definition. A Cogo Point Group will contain a range of Cogo Points. By using these objects, a Surface can be built using ranges of Cogo Points.

No validation exists between the PointGroupName and a Cogo Point Group. When a PointGroupName is added, AutoCAD Land Desktop does not check to see if the PointGroupName.Name actually exists as a Cogo Point Group.
<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Application</td>
<td>Modified</td>
</tr>
<tr>
<td>Delete</td>
<td>Count</td>
<td></td>
</tr>
<tr>
<td>Item</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The PointGroups collection represents all of the Point Groups in the project database.

**VBA object name:** AeccPointGroups

**Create using:** N/A

**Access via:** Project.PointGroups

Cogo PointGroups allow the user to work with a subset of Cogo points. This subset can also have the elevation, description, and name of the points overridden. This would be useful for when a point would need multiple elevations.

**Methods**
- Add
- Delete

**Properties**
- Application
- AutoSave
- Count

**Events**
- Modified
This object specifies the current AutoCAD Land Desktop settings.

**VBA object name:** AeccPreferences

**Create using:** N/A

**Access via:** Application.Preferences

The Preferences object holds all the options stored external to the drawing such as User Preferences. Options that reside in the drawing can be accessed through the DatabasePreferences object.

The Preferences object is divided into separate objects, with each representing a set of related options.

The Preferences object can be referenced from the Preferences property on the Application object.

See the Preferences object in the AutoCAD ActiveX and VBA Reference for information regarding additional Methods, Properties and Events provided through this object.
<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Application</td>
<td></td>
</tr>
<tr>
<td>None</td>
<td>Files</td>
<td>None</td>
</tr>
<tr>
<td>User</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Object

This object specifies the settings for Horizontal Alignments.

**VBA object name:** AeccPreferencesAlignment

**Create using:** N/A

**Access via:** PreferencesProject.PreferencesAlignment

**Methods**
- GetDouble
- GetInteger
- GetString
- SetDouble
- SetInteger

**Properties**
- Application
- Modified
SetString
This object specifies the settings for Points.

**VBA object name:** AeccPreferencesCogo

**Create using:** N/A

**Access via:** PreferencesProject.PreferencesCogo

**Methods**
- GetDouble
- GetInteger
- GetString
- SetDouble
- SetInteger
- SetString

**Properties**
- Application
- Modified

**Events**
PreferencesCrossSection

Object (Civil Engineering Feature)

This object specifies the settings for Alignment Cross Sections.

**VBA object name:** AeccPreferencesCrossSection

**Create using:** N/A

**Access via:** PreferencesProject.PreferencesCrossSection

**Methods**

- GetDouble
- GetInteger
- GetString
- SetDouble
- SetInteger
SetString
This object specifies the paths used by the AutoCAD Land Desktop.

**VBA object name:** AeccPreferencesFiles

**Create using:** N/A

**Access via:** Preferences.PreferencesFiles

This object specifies the paths used by the program. Some paths are established during installation and are exposed as read-only properties.

Paths are returned without a trailing backslash delimiter "\".

Changes to [BorderPath](#), [CivilDataFilesPath](#), [ContourStylesPath](#), [DrawingSetupPath](#), [FormatsPath](#), [LabelStylePath](#), [PreferencesPath](#), [ProjectPath](#), [PrototypePath](#), [SpeedTablesPath](#), [SymbolManagerPath](#) and [TempPath](#) will not take affect until you exit the program and restart.

See the [PreferencesFiles](#) object in the AutoCAD ActiveX and VBA Reference for information regarding additional Methods, Properties and Events provided through this object.
<table>
<thead>
<tr>
<th>Properties</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td></td>
</tr>
<tr>
<td>BorderPath</td>
<td></td>
</tr>
<tr>
<td>CivilDataFilesPath</td>
<td></td>
</tr>
<tr>
<td>ContourStylesPath</td>
<td></td>
</tr>
<tr>
<td>DataPath</td>
<td></td>
</tr>
<tr>
<td>DrawingSetupPath</td>
<td></td>
</tr>
<tr>
<td>FormatsPath</td>
<td></td>
</tr>
<tr>
<td>HelpPath</td>
<td></td>
</tr>
<tr>
<td>LabelStylePath</td>
<td></td>
</tr>
<tr>
<td>PreferencesPath</td>
<td></td>
</tr>
<tr>
<td>ProgramPath</td>
<td></td>
</tr>
<tr>
<td>ProjectPath</td>
<td></td>
</tr>
<tr>
<td>PrototypePath</td>
<td></td>
</tr>
<tr>
<td>SpeedTablesPath</td>
<td></td>
</tr>
<tr>
<td>SymbolManagerPath</td>
<td></td>
</tr>
<tr>
<td>SystemPath</td>
<td></td>
</tr>
<tr>
<td>TempPath</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Events</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>
PreferencesParcel

Object

This object specifies the settings for Parcels.

VBA object name: AeccPreferencesParcel

Create using: N/A

Access via: PreferencesProject.PreferencesParcel

Methods

GetInteger

GetString

Properties

Events

Application Modified

SetInteger

SetString
PreferencesProfile Object (Civil Engineering Feature)

This object specifies the settings for Alignment Vertical Profiles.

**VBA object name:** AeccPreferencesProfile

**Create using:** N/A

**Access via:** PreferencesProject.PreferencesProfile

**Methods**

- GetDouble
- GetInteger
- GetString
- SetDouble
- SetInteger
- SetString
Object

This object specifies the project settings.

**VBA object name:** AeccPreferencesProject

**Create using:** N/A

**Access via:** Project.Preferences

The PreferencesProject object holds all the options associated with external data such as Point Settings, Alignments, etc.

The PreferencesProject object can be referenced from the Preferences property on the Project object.

<table>
<thead>
<tr>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
</tr>
<tr>
<td>Alignment</td>
</tr>
</tbody>
</table>
Methods

None

Cogo

CrossSection (Civil Engineering Feature) Modified

Parcel

Profile (Civil Engineering Feature)

Surface
This object specifies the settings for Surfaces.

**VBA object name:** AeccPreferencesSurface

**Create using:** N/A

**Access via:** PreferencesProject.PreferencesSurface

**Methods**
- GetDouble
- GetInteger
- GetString
- SetDouble
- SetInteger

**Properties**
- Application
- Modified
SetString
PreferencesUser Object

This object specifies the options maintained on a per-user basis.

**VBA object name:** AeccPreferencesUser

**Create using:** N/A

**Access via:** Preferences.PreferencesUser

The User Preferences control program-wide preferences such as the AutoCAD overrides and the drawing setup method.

These settings are saved at the PreferencesPath in the file <AutoCAD login name>.dfm.

Changes to **OverrideNew**, **OverrideOpen** and **ShowStartupDialog** will not take affect until you exit the program and restart.

See the **PreferencesUser** object in the AutoCAD ActiveX and VBA Reference for information regarding additional Methods, Properties and Events provided through this object.
<table>
<thead>
<tr>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
</tr>
<tr>
<td>FirstTimeDrawingSetup</td>
</tr>
<tr>
<td>FirstTimeDrawingSetupFile</td>
</tr>
<tr>
<td>LastUsedDwg</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LastUsedDwgPath</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LastUsedProj</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>LastUsedProjPath</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>OverrideNew</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>OverrideOpen</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>ShowStartupDialog</th>
</tr>
</thead>
</table>
ProfileBlock Object
(Civil Engineering Feature)

The ProfileBlock object represents a profile in the current drawing.

VBA object name: AeccProfileBlock

Create using: N/A

Access via: ProfileBlocks.Item

Properties

  Application
  Coordinates
  CurveLabelIncrement
  DatumElevation
  Direction

Methods
Methods

- GetLayerName
- StationElevationToXy
- XyToStationElevation
- EGPPrecision
- EndingStation
- FGPPrecision
- Name
- StartingStation
- StationIncrement
- TangentLabelIncrement
- UpperRight
- VerticalScale

Events

- Modified
ProfileBlocks Collection
(Civil Engineering Feature)

The ProfileBlocks collection represents all of the profiles that are in the current drawing.

**VBA object name:** AeccProfileBlocks

**Create using:** N/A

**Access via:** Document.ProfileBlocks

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Item</td>
<td>Modified</td>
</tr>
<tr>
<td>Count</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


An AutoCAD Land Desktop project.

**VBA object name:** AeccProject

**Create using:** Projects.Add

**Access via:** Projects.Item Application.ActiveProjects

The **Project** object represents a project in the AutoCAD Land Desktop. Projects are used to manage and organize all the data for a job that you are working on. This data includes the project point file, alignment database, parcel database, surface database, drawing files, and so on.

AutoCAD Land Desktop works with only one project at a time. You can query certain properties of all Project objects in the **Projects** collection, but access to project-based data (such as **CogoPoints**) is only allowed in the **ActiveProject**. The ActiveProject.object has its **Active** property set to TRUE.
The collection of all AutoCAD Land Desktop projects on the network.

**VBA object name:** AeccProjects

**Create using:** N/A

**Access via:** Application.Projects

Use the Projects collection to find a particular project by name, author or keywords. You can also use the Projects collection to create a new project based on a prototype.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Application</td>
<td>Modified</td>
</tr>
<tr>
<td>Delete</td>
<td>Count</td>
<td></td>
</tr>
<tr>
<td>Item</td>
<td>ProjectPath</td>
<td></td>
</tr>
</tbody>
</table>
An AutoCAD Land Desktop project prototype.

**VBA object name:** AeccPrototype

**Create using:** Prototypes.Copy

**Access via:** Prototypes.Item

AutoCAD Land Desktop uses prototypes as a convenient way for you to maintain standard settings for your drawings. Whenever a new drawing is attached to a project, its default settings are copied from this prototype. The settings are copied to each drawing so that after a drawing is created, its settings can be modified independently of any of the other drawings in the project.
Name
Prototypes Collection

The collection of all AutoCAD Land Desktop project prototypes.

**VBA object name:** AeccPrototypes

**Create using:** N/A

**Access via:** Application.Prototypes

The collection of prototypes at the **PrototypePath**.

**Methods:**
- Copy
- Delete
- Count
- Item

**Properties:**
- Application

**Events:**
- Modified
The PVI object represents a Point of Vertical Intersection for a Finished Ground Profile.

**VBA object name:** AeccPVI

**Create using:** PVIs.Add

**Access via:** PVIs.Item

**Properties**
- Application

**Methods**
- CurveLength

**Events**
- Modified

- Elevation

- Station
The PVIs collection represents all of the Point of Vertical Intersection objects in the project.

**VBA object name:** AeccPVIs

**Create using:** N/A

**Access via:** FGProfile.PVIs

**Methods:**
- Add
- Delete
- Count
- Item

**Properties:**
- Application

**Events:**
- Modified
The StationEquation object represents a station equation for the alignment.

**VBA object name:** AeccStationEquation

**Create using:** StationEquations.Add

**Access via:** StationEquations.Item

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>StationAhead</td>
<td>Modified</td>
</tr>
<tr>
<td>None</td>
<td>StationBack</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The StationEquations collection represents all of the station equations for the Alignment.

**VBA object name:** AeccStationsEquations

**Create using:** N/A

**Access via:** Alignment.StationsEquations

**Methods**
- Add
- Delete
- Item
- RemoveAll

**Properties**
- Application
- Count

**Events**
- Modified
Superelevation Object
(Civil Engineering Feature)

The Superelevation object represents a single superelevation for an alignment.

**VBA object name:** AeccSuperelevation

**Create using:** N/A

**Access via:** Superelevations.Item

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>CurveCode</td>
<td>Modified</td>
</tr>
<tr>
<td>None</td>
<td>Station</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SuperelevationCode</td>
<td></td>
</tr>
</tbody>
</table>
Superelevations Collection (Civil Engineering Feature)

The Superelevations collection represents all of the superelevations for an alignment.

**VBA object name:** AeccSuperelevations

**Create using:** N/A

**Access via:** Alignment.Superelevations

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Count</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Events</th>
</tr>
</thead>
</table>
The Surface object represents a Surface in the project database.

**VBA object name:** AeccSurface

**Create using:** Surfaces.Add

**Access via:** Surfaces.Item

### Properties
- Application
- Area2D
- Area3D
- AverageGrade
- Description

### Methods
- Inputs
AddToAllElevations  LockType
AreaVolume  MaxElevation
Build  MaxFaceArea
FindAllConnectingEdges  MaxGrade
FindConnectingEdge  MeanElevation
FindPoint  MinElevation
GetBoundingBox  MinFaceArea
GetElevation  MinGrade
Import  Name
Paste  NumberOfFaces
SampleElevations  NumberOfPoints
SetBoundingBox  Outputs
  PointOnLineTolerance
  PointTolerance
  RevisionNumber
  Status
  Type
  Volume
The SurfaceInputs object represents all inputs for a Surface.

**VBA object name:** AeccSurfaceInputs

**Create using:** N/A

**Access via:** Surface.SurfaceInputs

**Properties**
- Application
- Boundaries

**Methods**
- BreakLines

**Events**
- Modified

**None**
- ContourItems
- DEMFiles
PointFiles

PointGroupNames
SurfaceOutputs Object

The SurfaceOutputs object represents all outputs for a Surface.

**VBA object name:** AeccSurfaceOutputs

**Create using:** N/A

**Access via:** Surface.SurfaceOutputs

**Properties**

- Application
- Edges
- ElevationContours
- Faces
- TinPoints
- WaterSheds

**Methods**

None

**Events**

- Modified
The Surfaces collection represents all of the Surface objects in the project database.

**VBA object name:** AeccSurfaces

**Create using:** N/A

**Access via:** Project.Surfaces

**Methods**

- Add
- Composite
- Copy
- Delete
- DifferenceGrid
- Item

**Properties**

- Application
- CurrentSurface

**Events**

- Count
- Modified
Rename
The TinPoint represents a calculated point for the Surface.

**VBA object name:** AeccTinPoint

**Create using:** N/A

**Access via:** TinPoints.Item

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Easting</td>
<td>Modified</td>
</tr>
<tr>
<td>None</td>
<td>Elevation</td>
<td>Northing</td>
</tr>
</tbody>
</table>
**TinPoints Collection**

The TinPoints represents all of the calculated points for the Surface.

**VBA object name:** AeccTinPoints

**Create using:** N/A

**Access via:** SurfaceOutputs.TinPoints

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Item</td>
<td>Modified</td>
</tr>
<tr>
<td>Count</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A series of methods provided for utility purposes.

**VBA object name:** AeccUtility

**Create using:** N/A

**Access via:** Document.Utility

See the Utility object in the AutoCAD ActiveX and VBA Reference for information regarding additional Methods, Properties and Events provided through this object.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>EastNorthToXy</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>XyToEastNorth</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
The WaterShed object represents a single watershed item for the WaterSheds collection.

**VBA object name**: AeccWaterShed

**Create using**: N/A

**Access via**: WaterSheds.Item

### Properties
- **Application**
- **Boundary**
- **Id**
- **OverflowPoints**
- **Tvne**

### Methods
- **DrainsInto**
- None

### Events
- **Modified**
The WaterSheds collection represents all of the WaterShed items for the Surface.

**VBA object name:** AeccWaterSheds

**Create using:** N/A

**Access via:** SurfaceOutputs.WaterSheds

### Properties
- **Application**

### Methods
- **Build**
  - **Count**
  - **ExceedBoth**

### Item
- **MinDepressionArea**
- **MinDepressionDepth**
A Example

Sub Example_A()

' This example returns the A value for the first Spiral found in the
' first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Spiral entity in the first Alignment."

' Find first Spiral in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kSpiral Then
        "The A value for the first Spiral in the alignment is: " & alignEnt.A
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "A Example"

End Sub
Sub Example_Active()

' This example returns the Active setting for the first Project
' in the collection
Dim proj As AeccProject
Set proj = AeccApplication.Projects.Item(0)

MsgBox "The Active value for the first Project in the collection is: " & proj.Active,
    vbInformation, "Active Example"

End Sub
Sub Example_ActiveDocument()

    ' This example returns name of the CurrentDocument.
    Dim activeDoc As AeccDocument
    Set activeDoc = AeccApplication.ActiveDocument
    MsgBox "The ActiveDocument is: " & activeDoc.Name, _
        vbInformation, "ActiveDocument Example"

End Sub
ActiveProject Example

Sub Example_ActiveProject()

' This example returns the name of the ActiveProject.
Dim activeProj As AeccProject
Set activeProj = AeccApplication.ActiveProject

MsgBox "The ActiveProject is: " & activeProj.Name, _
vbInformation, "ActiveProject Example"

End Sub
AD1 Example

Sub Example_AD1()

' This example returns the AD1 value for the first Spiral found in the first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Spiral entity in the first Alignment."

' Find first Spiral in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kSpiral Then
        alignMsg; = "The AD1 value for the first Spiral in the alignment is: " & al
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "AD1 Example"

End Sub
Sub Example_AD2()

' This example returns the AD2 value for the first Spiral found in the
' first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ACTiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Spiral entity in the first Alignment."

' Find first Spiral in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kSpiral Then
        alignMsg &= "The AD2 value for the first Spiral in the alignment is: " & alignEnt.AD2
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "AD2 Example"

End Sub
Examples:

- Alignments
- Boundaries
- BreakLines
- CogoPoints
- ContourItems
- DEMFiles
- DescriptionKeyFile
- DescriptionKeyFiles
- EGProfiles (Civil Engineering Feature)
- FGProfiles (Civil Engineering Feature)
- Parcels
- PointFiles
- PointGroupNames
- PointGroups
- Projects
- PVIs (Civil Engineering Feature)
- StationEquations
- Surfaces
Sub Example_Add_Alignments()

' This example adds an Alignment made up of a tangent, 
' curve, and spiral entities.
Dim aligns As AeccAlignments
Dim align As AeccAlignment
Set aligns = AeccApplication.ActiveProject.Alignments

' Add an Alignment named "Example Alignment" and starting at Station 50.0
Set align = aligns.Add("Example Alignment", 50#)

' Add a tangent, curve, and spiral to the alignment
Dim tangent As AeccAlignTangent
Dim curve As AeccAlignCurve
Dim spiral As AeccAlignSpiral

Set tangent = align.AddTangent(0#, 0#, 150#, 0#)
Set spiral = align.AddSpiral(150#, 0#, 388.069176379758, -5.83082052087824E-14, 250#, 100#,
Set curve = align.AddCurve(250, 100#, 320.966940499621, 197.983470249737

MsgBox "The total number of entities in the Alignment is: " & align.AlignEntities.count,
vbInformation, "Add Example"

End Sub

Sub Example_Add_Boundaries()

' This example starts by displays the initial count of the Boundaries
' for the first surface in the collection. A new visible Boundary is added
' and the count is redispays. Finally, the new Boundary is deleted. The count is
' displayed again, showing the deletion.
Dim surf As AeccSurface
Dim bounds As AeccBoundaries
Dim Bound As AeccBoundary
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set bounds = surf.Inputs.Boundaries
Sub Example_Add_BreakLines()

    ' This example starts by displaying the initial count of the BreakLines
    ' for the first surface in the collection. A new BreakLines is added
    ' and the count is redisplayed. Finally, the new file is deleted. The count is
    ' displayed again, showing the deletion.
    Dim surf As AeccSurface
    Dim brkLines As AeccBreakLines

End Sub
Dim brkLine As AeccBreakLine
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set brkLines = surf.Inputs.BreakLines

MsgBox "The BreakLines Count is: " & brkLines.count, vbInformation, "Add Example"

' Initialize variables
Dim pnts(0 To 14) As Double
Dim count As Integer
Dim index As Integer
Dim pnt As Variant

index = 0

' Add a new BreakLine based on five selected points
For count = 1 To 5
    pnt = ThisDrawing.Utility.GetPoint(, "Select point" + Str(count) + " of BreakLine")
    pnts(index) = pnt(0): pnts(index + 1) = pnt(1): pnts(index + 2) = pnt(2)
    index = index + 3
Next count

Set brkLine = brkLines.Add(pnts, "NewBreakLine")

MsgBox "The BreakLines Count after the add is: " & brkLines.count, vbInformation

' Delete the new BreakLines
brkLines.Delete brkLine.Id

MsgBox "The BreakLines Count after the delete is " & brkLines.count, vbInformation

End Sub

Sub Example_Add_CogoPoints()

' This function adds a new CogoPoint to the CogoPoints collection.
Dim cogoPnts As AeccCogoPoints
Dim newCogoPnt As AeccCogoPoint
Set cogoPnts = AeccApplication.ActiveProject.CogoPoints

' Get a point
Dim newPnt As Variant
newPnt = ThisDrawing.Utility.GetPoint(, "Select point location: ")

' Add CogoPoint
Set newCogoPnt = cogoPnts.Add(newPnt, kCoordinateFormatXYZ)

MsgBox "The Number for the new CogoPoint is: " & newCogoPnt.Number, vbInformation, "Add Example"

End Sub

Sub Example_Add_ContourItems()

' This example Adds a Contour, with an elevation of 100,
' to the first collection in ContourItems.
Dim surf As AeccSurface
Dim conts As AeccContourItems
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set conts = surf.Inputs.ContourItems

' Display the number of ContourItems in the collection before the add
MsgBox "The number of ContourItems in the collection is: " & conts.count, vbInformation

' Initialize variables
Dim pnts(0 To 29) As Double
Dim count As Integer
Dim index As Integer
Dim pnt As Variant

index = 0

'Define a new contour based upon 10 entered points
For count = 1 To 10
    pnt = ThisDrawing.Utility.GetPoint(, "Select point" + str(count) + " of Contour")
    pnts(index) = pnt(0): pnts(index + 1) = pnt(1): pnts(index + 2) = 100#
End For
```vbnet
index = index + 3
Next count

' Add Contour to the collection
conts.Add Points

' Display the number of ContourItems in the collection after the add
MsgBox "The number of ContourItems in the collection is: " & conts.count, vbInformation, "Add Example"
End Sub

Sub Example_Add_DEMFiles()

' This example starts by displays the initial count of DEMFiles
' for the first surface. A new DEMfile is added and the count is redispalyes.
' Finally, the new file is deleted. The count is displayed again, showing
' the deletion.
Dim surf As AeccSurface
Dim DEMFiles As AeccDEMFiles
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set DEMFiles = surf.Inputs.DEMFiles

MsgBox "The DEMFiles Count is: " & DEMFiles.Count, vbInformation, "Add Example"

' Get the DEMFile name and format the prompt
Dim DEMName As String
Dim prompt As String

prompt = "Enter the name of the DEMFile for surface " & surf.Name & ": "
DEMName = ThisDrawing.Utility.GetString(False, prompt)

' Add a new DEMFile
DEMFiles.Add DEMName

MsgBox "The DEMFiles Count after the add is: " & DEMFiles.Count, vbInformation, "Add Example"

' Delete the new DEMFile
```
DEMFiles.Delete DEMName

MsgBox "The DEMFiles Count after the delete is " & DEMFiles.Count, vbInformation

End Sub

Sub Example_Add_DescriptionKeyFile()

' This example starts by displaying the initial count of DescriptionKeys
' in the DEFAULT DescriptionKey file. A new DescriptionKey is added
' and the count is redisplayed. Finally, the new key is deleted. The
' count is displayed again, showing the deletion.
Dim dkeyFile As AeccDescriptionKeyFile
Set dkeyFile = AeccApplication.ActiveProject.DescriptionKeyFiles.Item("DEFAULT")

MsgBox "The DescriptionKey Count is " & dkeyFile.count, vbInformation, "Add Example"

' Add a new DescriptionKey
dkeyFile.Add "New"

MsgBox "The DescriptionKey Count is " & dkeyFile.count, vbInformation, "Add Example"

' Delete the new DescriptionKey
dkeyFile.Delete "New"

MsgBox "The DescriptionKey Count is " & dkeyFile.count, vbInformation, "Add Example"

End Sub

Sub Example_Add_DescriptionKeyFiles()

; This example starts by displaying the initial count of DescriptionKeyFiles
; on the system. A new DescriptionKeyFile is added and the count is redisplayed
; Finally, the new file is deleted. The count is displayed again, showing
; the deletion.
Dim dKeyFiles As AeccDescriptionKeyFiles
Set dKeyFiles = AeccApplication.ActiveProject.DescriptionKeyFiles

MsgBox "The DescriptionKeyFiles Count is " & dKeyFiles.count, vbInformation

' Add a new DescriptionKeyFile
dKeyFiles.Add "NewFile"

MsgBox "The DescriptionKeyFiles Count is " & dKeyFiles.count, vbInformation

' Delete the new DescriptionKeyFile
dKeyFiles.Delete "NewFile"

MsgBox "The DescriptionKeyFiles Count is " & dKeyFiles.count, vbInformation

End Sub

Sub Example_Add_EGProfiles()

' This example gets the surface name from the user and adds a new existing ground profile for the first alignment in the collection.
Dim align As AeccAlignment
Dim newEGProf As AeccEGProfile
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

'Get the surface name and format the prompt
Dim surfName As String
Dim prompt As String

prompt = "Enter the name of the surface for alignment " & align.Name & ": "
surfName = ThisDrawing.Utility.GetString(False, prompt)

'Add the new existing ground profile
Set newEGProf = align.EGProfiles.Add(kEgCenter, surfName)

MsgBox "The first station for the new existing ground profile is: " _ &
     & newEGProf.StationElevations(0), vbInformation, "Add Example"

End Sub
Sub Example_Add_FGProfiles()

    ' This example adds a center type finished ground profile
    ' to the first alignment in the collection.
    Dim align As AeccAlignment
    Dim newFGProf As AeccFGProfile
    Set align = AeccApplication.ActiveProject.Alignments.Item(0)

    'Add the new finished ground profile
    Set newFGProf = align.FGProfiles.Add(kFgCenter)

    MsgBox "The type for the new existing ground profile is: "
        & newFGProf.Type, vbInformation, "Add Example"

End Sub

Sub Example_Add_Parcels()

    ' This example starts by displays the initial count of Parcels
    ' A new Parcel is added and the count is redisplayed. Finally,
    ' the new Parcel is deleted. The count is displayed again, showing
    ' the deletion.

    Dim parcels As AeccParcels
    Set parcels = AeccApplication.ActiveProject.Parcels

    MsgBox "The Parcel Count is: " & parcels.Count, vbInformation, "Add Examụp" 

    ' Add a new Parcel
    parcels.Add "NewParcel"

    MsgBox "The Parcel Count is: " & parcels.Count, vbInformation, "Add Examụp"

    ' Delete the new Parcel
    parcels.Delete "NewParcel"

MsgBox "The Parcel Count is: " & parcels.Count, vbInformation, "Add Example"
End Sub

Sub Example_Add_PointFiles()

' This example starts by displaying the initial count of PointFiles
' on the system. A new Pointfile is added and the count is redispays.
' Finally, the new file is deleted. The count is displayed again, showing
' the deletion.
Dim surf As AeccSurface
Dim pntFiles As AeccPointFiles
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set pntFiles = surf.Inputs.PointFiles

MsgBox "The PointFiles Count is: " & pntFiles.Count, vbInformation, "Add Example"

' Add a new PointFile
pntFiles.Add "NewPointFile"

MsgBox "The PointFiles Count after the add is: " & pntFiles.Count, vbInformation, "Add Example"

' Delete the new PointFile
pntFiles.Delete "NewPointFile"

MsgBox "The PointFiles Count after the delete is: " & pntFiles.Count, vbInformation, "Add Example"
End Sub

Sub Example_Add_PointGroupNames()

' This example starts by displays the initial count of the PointGroupNames
' for the first surface in the collection. A new PointGroupName is added
' and the count is redispays. Finally, the new file is deleted. The count is
' displayed again, showing the deletion.
Dim surf As AeccSurface
Dim pntGrpNames As AeccPointGroupNames

Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set pntGrpNames = surf.Inputs.PointGroupNames

MsgBox "The PointGroupNames Count is: " & pntGrpNames.Count, vbInformation, "Add Example"

' Add a new PointGroupName
pntGrpNames.Add "NewPointGroupName"

MsgBox "The PointGroupNames Count after the add is: " & pntGrpNames.Count, vbInformation, "Add Example"

' Delete the new PointGroupName
pntGrpNames.Delete "NewPointGroupName"

MsgBox "The PointGroupNames Count after the delete is " & pntGrpNames.Count, vbInformation, "Add Example"

End Sub

Sub Example_Add_PointGroups()

' This function adds a new PointGroup named "New Group".
Dim pntGrps As AeccPointGroups
Dim newPntGrp As AeccPointGroup
Set pntGrps = AeccApplication.ActiveProject.PointGroups

' Show the number of alignments in the project
MsgBox "The initial Count of PointGroups is: " & pntGrps.Count, vbInformation

' Add PointGroup
Set newPntGrp = pntGrps.Add("Example Group", "1-10, 20-30")

' Show the number of alignments in the project after the Add
MsgBox "The Count of PointGroups is: " & pntGrps.Count, vbInformation, "Add Example"

End Sub

Sub Example_Add_Projects()
'This example creates a new project.
Dim projs As AeccProjects
Set projs = AeccApplication.Projects

'Show the number of projects
MsgBox "The initial Count of Projects is: " & projs.Count, vbInformation, "Add Example"
projs.Add "New Project", "Default (Feet)"

'Show the number of projects after the Add
MsgBox "The Count of Projects is: " & projs.Count, vbInformation, "Add Example"
End Sub

Sub Example_Add_PVIs()

'This example adds a PVI in the first finished ground profile of the first alignment in the collection.
Dim align As AeccAlignment
Dim FGProf As AeccFGProfile
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set FGProf = align.FGProfiles.Item(0)

MsgBox "The number of PVIs in the finished ground profiles is: " _ & FGProf.PVIs.Count, vbInformation, "Add Example"

'Get the station, elevation, and curve length
Dim station As Double
Dim elevation As Double
Dim curvelength As Double
station = ThisDrawing.Utility.GetReal("Enter a station on finished grade profile:
latitude = ThisDrawing.Utility.GetReal("Enter an elevation on finished grade profile:
station = ThisDrawing.Utility.GetReal("Enter a curve length on finished grade profile:

'Add the PVI
FGProf.PVIs.Add station, elevation, curvelength
Sub Example_Add_StationEquations()

' This example adds a StationEquation to
' the first alignment in the collection.
Dim aligns As AeccAlignments
Dim align As AeccAlignment
Set aligns = AeccApplication.ActiveProject.Alignments

' Get the first alignment
Set align = aligns.Item(0)

' Set the first alignment current.
aligns.CurrentAlignment = align.Name

Dim staEqus As AeccStationEquations
Dim staEqu As AeccStationEquation
Set staEqus = align.StationEquations

' Show the StationEquation count for the first alignment
MsgBox "The StationEquation count for the first alignment is: " & staEqus.count,
vbInformation, "Add Example"

' Add a new StationEquations
Set staEqu = staEqus.Add(100, 50, kIncreasing)

' Show the StationEquation count after the add
MsgBox "The StationEquation count for the first alignment is: " & staEqus.count,
vbInformation, "Add Example"

End Sub

Sub Example_Add_Surfaces()
'This example creates a new Surface and adds it to the Surfaces collection

Dim surfs As AeccSurfaces
Set surfs = AeccApplication.ActiveProject.Surfaces

Surfs.Add "New Surface"

MsgBox "The Number of Surfaces has increased to " & surfs.Count, vbInformation

End Sub
AddCurve Example

Examples:

1 Alignment

1 Parcel

Sub Example_AddCurve_Alignment()

' This example adds an Alignment made up of a tangent, curve, and spiral entities.
Dim aligns As AeccAlignments
Dim align As AeccAlignment
Set aligns = AeccApplication.ActiveProject.Alignments

' Add an Alignment named "Example Alignment" and starting at Station 50.0
Set align = aligns.Add("Example Alignment", 50#)

' Add a tangent, curve, and spiral to the alignment
Dim tangent As AeccAlignTangent
Dim curve As AeccAlignCurve
Dim spiral As AeccAlignSpiral

Set tangent = align.AddTangent(0#, 0#, 150#, 0#)
Set spiral = align.AddSpiral(150#, 0#, 388.069176379758, -5.83082052087824E-14, 250#, 100#, 0, 0, kClothoid)
Set curve = align.AddCurve(250, 100#, 320.966940499621, 197.983470249737, 200#, 200#,
False)

MsgBox "The total number of entities in the Alignment is: " & align.AlignEntities.count, vbInformation, "AddCurve Example"

End Sub

Sub Example_AddCurve_Parcel()

' This example adds a Parcel made up of lines and an arc.
' The parcel is then imported into the drawing.
Dim parcels As AeccParcels
Dim parcel As AeccParcel
Set parcels = AeccApplication.ActiveProject.parcels

' Add a new Parcel
Set parcel = parcels.Add("New Parcel")

' Add lines and a curve to the Parcel
parcel.AddLine 50#, 50#, 150#, 50#
parcel.AddLine 150#, 50#, 150#, 200#
parcel.AddCurve 150#, 200#, 100#, 200#, 50#, 200#, True
parcel.AddLine 50#, 200#, 50#, 50#

' Import the new Parcel
parcel.Import

MsgBox "The total number of entities in the parcel is: " & parcel.ParcelEntities.Count, vbInformation, "AddCurve Example"

End Sub
AddLabel Example

Sub Example_AddLabelAt()

' This example adds a label to a Contour object by selecting the ' contour

On Error Resume Next

' Delete existing SelectionSet
ThisDrawing.SelectionSets("SSet").Delete

' Create the selection set based on a point selection ' and filter for Contour objects
Dim ssetObj As AcadSelectionSet
Set ssetObj = ThisDrawing.SelectionSets.Add("SSet")

Dim mode As Integer
Dim gpCode(0) As Integer
Dim dataValue(0) As Variant

gpCode(0) = 0
dataValue(0) = "AECC_CONTOUR"

Dim groupCode As Variant
Dim dataCode As Variant
Dim returnPnt As Variant

groupCode = gpCode
dataCode = dataValue
returnPnt = ThisDrawing.Utility.GetPoint(, "Enter a point on a contour line: ")
ssetObj.SelectAtPoint returnPnt, groupCode, dataCode

Dim Ent As AeccContour
Set Ent = ssetObj.Item(0)
Ent.AddLabelAt returnPnt(0), returnPnt(1)
End Sub
AddLine Example

Sub Example_AddLine()

    ' This example adds a Parcel made up of lines and an arc.
    ' The parcel is then imported into the drawing.
    Dim parcels As AeccParcels
    Dim parcel As AeccParcel
    Set parcels = AeccApplication.ActiveProject.parcels

    ' Add a new Parcel
    Set parcel = parcels.Add("New Parcel")

    ' Add lines and a curve to the Parcel
    parcel.AddLine 50#, 50#, 150#, 50#
    parcel.AddLine 150#, 50#, 150#, 200#
    parcel.AddCurve 150#, 200#, 100#, 200#, 50#, 200#, True
    parcel.AddLine 50#, 200#, 50#, 50#

    ' Import the new Parcel
    parcel.Import

    MsgBox "The total number of entities in the parcel is: " & parcel.ParcelEntities.Count, vbInformation, "AddLine Example"

End Sub
AddSpiral Example

Sub Example_AddSpiral()

' This example adds an Alignment made up of a tangent, ' curve, and spiral entities.
Dim aligns As AeccAlignments
Dim align As AeccAlignment
Set aligns = AeccApplication.ActiveProject.Alignments

' Add an Alignment named "Example Alignment" and starting at Station 50.0
Set align = aligns.Add("Example Alignment", 50#)

' Add a tangent, curve, and spiral to the alignment
Dim tangent As AeccAlignTangent
Dim curve As AeccAlignCurve
Dim spiral As AeccAlignSpiral

Set tangent = align.AddTangent(0#, 0#, 150#, 0#)
Set spiral = align.AddSpiral(150#, 0#, 388.069176379758, -5.83082052087824E-14, 250#, 100#, 0, 0, kClothoid)
Set curve = align.AddCurve(250, 100#, 320.966940499621, 197.983470249737, 200#, 200#, False)

MsgBox "The total number of entities in the Alignment is: " & align.AlignEnti

End Sub
Sub Example_AddTangent()

' This example adds an Alignment made up of a tangent, ' curve, and spiral entities.
Dim aligns As AeccAlignments
Dim align As AeccAlignment
Set aligns = AeccApplication.ActiveProject.Alignments

' Add an Alignment named "Example Alignment" and starting at Station 50.0
Set align = aligns.Add("Example Alignment", 50#)

' Add a tangent, curve, and spiral to the alignment
Dim tangent As AeccAlignTangent
Dim curve As AeccAlignCurve
Dim spiral As AeccAlignSpiral

Set tangent = align.AddTangent(0#, 0#, 150#, 0#)
Set spiral = align.AddSpiral(150#, 0#, 388.069176379758, -5.83082052087824E-14, 250#, 100#, 0, 0, kClothoid)
Set curve = align.AddCurve(250, 100#, 320.966940499621, 197.983470249737, 200#, 200#, False)

MsgBox "The total number of entities in the Alignment is: " & align.AlignEntities.count, vbInformation, "AddTangent Example"
End Sub
Sub Example_AddToAllElevations()

' This example uses AddToAllElevations to increase the elevation by 20.0 for
' the first surface in the collection.
Dim surf As AeccSurface
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

MsgBox "The MeanElevation for the first surface before elevation increase is:"
    surf.MeanElevation;, vbInformation, "AddToAllElevations Example"

surf.AddToAllElevations 20#
surf.Build

MsgBox "The MeanElevation for the first surface after elevation increase is: "
    surf.MeanElevation;, vbInformation, "AddToAllElevations Example"

End Sub
Sub Example_AlgEntities()

' This example gets the count of AlignEntities for the
' first alignment in the collection.
Dim align As AeccAlignment
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Dim count As Integer

' Get the number of entities
count = align.AlignEntities.count

If count = 0 Then
    MsgBox; "The first Alignment in the collection has no entities."
        vbInformation; "AlignEntities Example"
ElseIf count = 1 Then
    MsgBox; "The first Alignment in the collection has 1 entity."
        vbInformation; "AlignEntities Example"
ElseIf count > 1 Then
    MsgBox; "The first Alignment in the collection has " & count & " entities."
        vbInformation; "AlignEntities Example"
End If

End Sub
Alignment Example

Sub Example_Alignment()

    ' This example returns the PreferencesAlignment object that 
    ' is used to access the StationLayer property
    Dim alignPref As AeccPreferencesAlignment
    Set alignPref = AeccApplication.ActiveProject.preferences.Alignment

    MsgBox "The Alignment preferences for StationLayer is: " & alignPref.GetString(kStationLayer)

End Sub
Sub Example_AlignmentFromObjectID()

' This example returns the AlignmentFromObjectID for a selected alignment.
Dim aligns As AeccAlignments
Set aligns = AeccApplication.ActiveProject.Alignments

Dim alignObj As AcadObject
Dim basePnt As Variant
Dim alignID As Long
Dim alignNames As Variant

On Error Resume Next

' The following example prompts for a selection from the user
RETRY;
ThisDrawing.Utility.GetEntity alignObj, basePnt, "Select an object"

If Err <> 0 Then
   Err.Clear
   MsgBox "Good Bye.", vbInformation, "AlignmentFromObjectID Example"
   Exit Sub
Else
   alignID = alignObj.ObjectID
   alignNames = aligns.AlignmentFromObjectID(alignID)
   MsgBox "The selected alignment is named: " & alignNames(0), _
      vbInformation, "AlignmentFromObjectID Example"
End If

GoTo RETRY

End Sub
Sub Example_Alignments()

  ' This example returns the tick increment for Alignments Preferences
  ' in the current project.
  Dim prefPrj As AeccPreferencesProject
  Set prefPrj = AeccApplication.ActiveProject.Preferences

  MsgBox "The tick increment for Alignments Preferences in the current Project
         , vbInformation, "Alignments Example"

End Sub
Sub Example_AngularAzimuth()

' This example returns the AngularAzimuth setting for the current drawing.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.Preferences

' Convert the constant to a string.
Dim Style As String
Dim AngAzm As Long

' Determine AngularAzimuth setting
AngAzm = dbPref.AngularAzimuth
Select Case AngAzm
Case 0
    Style = "bearings."
Case 1
    Style = "north azimuths."
Case 2
    Style = "south azimuths."
End Select

MsgBox "The current value for AngularAzimuth is " & Style, vbInformation, "AngularAzimuth Example"

End Sub
Sub Example_AngularDisplayFormat()

    ' This example returns the AngularDisplayFormat setting for the current drawing.
    Dim dbPref As AeccDatabasePreferences
    Set dbPref = AeccApplication.ActiveDocument.Preferences

    ' Convert the constant to a string.
    Dim strUnits As String
    If dbPref.AngularDisplayFormat = 1 Then
        strUnits = "degrees."
    Else
        strUnits = "grads."
    End If

    MsgBox "The current value for AngularDisplayFormat is " & strUnits, _
        vbInformation, "AngularDisplayFormat Example"

End Sub
Sub Example_AngularPrecision()

' This example returns the AngularPrecision setting for the current drawing.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.Preferences

MsgBox "The current value for AngularPrecision is " & dbPref.AngularPrecision, vbInformation, "AngularPrecision Example"

End Sub
Sub Example_Application()

' This example uses the AeccApplication property to navigate to the top of
' the object hierarchy to compare the current project with the last used project.
Dim proj As AeccProject
Set proj = AeccApplication.ActiveProject

MsgBox "The current project is " & proj.Application.ActiveProject.Name & " and the last used project is " & _
vbInformation, "Application Example"

End Sub
Area Example

Sub Example_Area()

' This example returns the Area for the first parcel in the collection.
Dim parcel As AeccParcel
Set parcel = AeccApplication.ActiveProject.parcels.Item(0)

MsgBox "The Area for the first Parcel in the collection is: ", _
    & Format(parcel.Area, "0.00"), vbInformation, "Area Example"

End Sub
**Area2D Example**

**Examples:**

1. **Face**

1. **Surface**

---

**Sub** Example_Area2D_Face()

' This example returns the Area2D value for the first Face in the collection.
Dim face As AeccFace
Dim surf As AeccSurface
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set face = surf.Outputs.Faces.Item(0)

MsgBox "The Area2D value is " & face.Area2D, vbInformation, "Area2D Example"

End Sub

---

**Sub** Example_Area2D_Surface()

' This example returns the Area2D for the first surface in the collection.
Dim surf As AeccSurface
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

MsgBox "The Area2D for the first surface is: " & surf.Area2D, vbInformation,

End Sub
Area3D Example

Examples:

1 Face

1 Surface

Sub Example_Area3D_Face()

' This example returns the Area3D value for the
' first Face in the collection.
Dim face As AeccFace
Dim surf As AeccSurface
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set face = surf.Outputs.Faces.Item(0)

MsgBox "The Area3D value is " & face.Area3D, vbInformation, "Area3D Example"
End Sub

Sub Example_Area3D_Surface()

' This example returns the Area3D for the first surface in the collection.
Dim surf As AeccSurface
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

MsgBox "The Area3D for the first surface is " & surf.Area3D, vbInformation,
End Sub
Sub Example_AreaDisplayUnit()

' This example returns the AreaDisplayUnit setting for the current drawing.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.preferences

' Convert the area display unit to a string.
Dim unit As String

Select Case dbPref.AreaDisplayUnit
  Case aecUnitSquareInch
    unit = "square inch."
  Case aecUnitSquareFoot
    unit = "square foot."
  Case aecUnitSquareYard
    unit = "square yard."
  Case aecUnitSquareMil
    unit = "square millimeters."
  Case aecUnitSquareCentimeters
    unit = "square centimeters."
  Case aecUnitSquareDecimeters
    unit = "square decimeters."
  Case aecUnitSquareMeters
    unit = "square meters."
End Select

MsgBox "The current value for AreaDisplayUnit is " & unit, vbInformation, "\nEnd Sub
Sub Example_AreaPrecision()

' This example displays the AreaPrecision setting for the current drawing.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.preferences

MsgBox "The current value for AreaPrecision is: " & dbPref.AreaPrecision, vbInformation

End Sub
Sub Example_AreaSuffix()

    ' This example displays the AreaSuffix setting for the current drawing.
    Dim dbPref As AeccDatabasePreferences
    Set dbPref = AeccApplication.ActiveDocument.preferences

    MsgBox "The current value for AreaSuffix is: " & dbPref.AreaSuffix, _
        vbInformation, "AreaSuffix Example"

End Sub
Sub Example_AreaVolume()

' This example returns the Cut, Fill and Net Volumes of the first surface in the
Dim surf As AeccSurface
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

Dim count as Long
Dim index as Long
Dim cut As Double
Dim fill As Double
Dim tolerance As Double
Dim net As Double
Dim pnt as Variant
Dim areaPnts(0 To 14) As Double

index = 0

' Define a area based upon 5 entered points
For count = 1 To 5
  pnt = ThisDrawing.Utility.GetPoint(, "Select point" + str(count) + " for area:
  areaPnts(index) = pnt(0): areaPnts(index + 1) = pnt(1): areaPnts(index + 2) =
  index = index + 3
Next count

' Get volume data
surf.AreaVolume 1.0, areaPnts, cut, fill, net

MsgBox "The AreaVolume data for the first surface is:" & vbCrLf & _
  " Cut: " & cut & vbCrLf & _
  " Fill: " & fill & vbCrLf & _
  " Net: " & net, vbInformation, "AreaVolume Example"

End Sub
Sub Example_ArrayToPointString()

' This example returns a string of point numbers in CogoPoints format.
Dim cogoPnts As AeccCogoPoints
Set cogoPnts = AeccApplication.ActiveProject.CogoPoints

Dim pntString As String
Dim pntArray(0 To 8) As Long

' Create an array with the point numbers 1,4,5,6,8,40,10,7,32
pntArray(0) = 1: pntArray(1) = 4: pntArray(2) = 5
pntArray(3) = 6: pntArray(4) = 8: pntArray(5) = 40
pntArray(6) = 10: pntArray(7) = 7: pntArray(8) = 32

' Convert array to string
pntString = cogoPnts.ArrayToPointString(pntArray)

MsgBox "The CogoPoints point number string is " & pntString,
      vbInformation, "ArrayToPointString Example"

End Sub
Author Example

Sub Example_Author()

    ' This example returns the Author setting for the first Project
    ' in the collection
    Dim proj As AeccProject
    Set proj = AeccApplication.Projects.Item(0)

    MsgBox "The Author value for the first Project in the collection is: " & proj.Author,
    vbInformation, "Author Example"

End Sub
AutoSave Example

Examples:

- Alignments
- CogoPoints
- DescriptionKeyFiles
- PointGroups

---

Sub Example_AutoSave_Alignments()

' This example changes the Description and StartingStation
' for the first alignment in the collection.
Dim aligns As AeccAlignments
Dim align As AeccAlignment
Set aligns = AeccApplication.ActiveProject.Alignments
Set align = aligns.Item(0)

' Make the first alignment in the collection current
aligns.CurrentAlignment = align.Name

' Set AutoSave to FALSE to prevent writing the changes
' until they are all made
aligns.AutoSave = False

' Make changes
align.Description = "New Description"
align.StartingStation = 20#

' Save changes
align.Save

' Reset AutoSave to the default state of TRUE
aligns.AutoSave = True
Sub Example_AutoSave_CogoPoints()

' This example changes the Northing, Easting, and Elevation
' for the first CogoPoint in the collection.
Dim cogoPnts As AeccCogoPoints
Dim cogoPnt As AeccCogoPoint
Set cogoPnts = AeccApplication.ActiveProject.CogoPoints
Set cogoPnt = cogoPnts.Item(0)

' Set AutoSave to FALSE to prevent writing the changes
' until they are all made
CogoPnts.AutoSave = False

' Make changes
CogoPnt.Northing = 100#
CogoPnt.Easting = 100#
CogoPnt.Elevation = 150#

' Save changes
CogoPnt.Save

' Reset AutoSave to the default state of TRUE
CogoPnts.AutoSave = True

MsgBox "The following changes were made to CogoPoint " & CogoPnt.Number
  " Northing: " & CogoPnt.Northing & vbCrLf & _
  " Easting: " & CogoPnt.Easting & vbCrLf & _
  " Elevation: " & CogoPnt.Elevation, vbInformation, "AutoSave Example"

End Sub
Sub Example_AutoSave_DescriptionKeyFiles()

    ' This example changes the layer settings for the first DescriptionKey in the collection.
    Dim dKeyFiles As AeccDescriptionKeyFiles
    Dim dKeyFile As AeccDescriptionKeyFile
    Dim dKey As AeccDescriptionKey
    Set dKeyFiles = AeccApplication.ActiveProject.DescriptionKeyFiles
    Set dKeyFile = dKeyFiles.Item(0)
    Set dKey = dKeyFile.Item(0)

    ' Set AutoSave to FALSE to prevent writing the changes until they are all made
    dKeyFiles.AutoSave = False

    ' Make changes
    dKey.DescriptionLayer = "dKey Desc"
    dKey.SymbolLayer = "dKey Symb"

    ' Save changes
    dKey.Save

    ' Reset AutoSave to the default state of TRUE
    dKeyFiles.AutoSave = True

    MsgBox "The DescriptionLayer is: " & dKey.DescriptionLayer & vbCrLf & _
        "The SymbolLayer is: " & dKey.SymbolLayer, vbInformation, "AutoSave Example"

End Sub

Sub Example_AutoSave_PointGroups()

    ' This example changes the Description and Elevation for the first PointGroup in the collection.
    Dim pntGrps As AeccPointGroups
    Dim pntGrp As AeccPointGroup
Set pntGrps = AeccApplication.ActiveProject.PointGroups
Set pntGrp = pntGrps.Item(0)

' Set AutoSave to FALSE to prevent writing the changes
' until they are all made
pntGrps.AutoSave = False

' Make changes
pntGrp.Description = "New Description"
pntGrp.Elevation = 100#

' Save changes
pntGrp.Save

' Reset AutoSave to the default state of TRUE
pntGrps.AutoSave = True

MsgBox "The first PointGroup Description is: " & pntGrp.Description & vbCrLf
    "The first PointGroup Elevation is: " & pntGrp.Elevation, vbInformation, "AutoSave Example"

End Sub
Sub Example_AverageGrade()

' This example returns the Average Grade for the first surface in the collection.
Dim surf As AeccSurface
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

MsgBox "The AverageGrade for the first surface is: " & Format(surf.AverageC
vbInformation, "AverageGrade Example"

End Sub
Sub Example_BasePoint()

' This example displays the BasePoint setting for the current drawing.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.preferences

Dim varPoint As Variant
Dim point(0 To 2) As Double
Dim basePoint As String

' Get the base point
varPoint = dbPref.BasePoint
point(0) = varPoint(0)
point(1) = varPoint(1)
point(2) = varPoint(2)

' Format base point to a string
basePoint = Format(point(0)) + "", " + Format(point(1)) + ", " + Format(point(2)
MsgBox "The current value for BasePoint is: " & basePoint, vbInformation, "E

End Sub
Sub Example_BasePointNE()

' This example displays the BasePointNE setting for the current drawing.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.Preferences

Dim varPoint As Variant
Dim point(0 To 2) As Double
Dim basePointNE As String

' Get the base point
varPoint = dbPref.basePointNE
point(0) = varpoint(0)
point(1) = varpoint(1)
point(2) = varpoint(2)

' Format base point to a string
basePointNE = Format(point(0)) + "", " + Format(point(1)) + ", " + Format(point(2)) + ""

MsgBox "The current value for BasePointNE is: " & basePointNE, _
    vbInformation, "BasePointNE Example"

End Sub
BeginCondition Example

Sub Example_BeginCondition()

' This example returns the BeginCondition for the first Spiral found in the first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Spiral entity in the first Alignment."

' Find first Spiral in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kSpiral Then
        alignMsg = "The BeginCondition for the first Spiral in the alignment is: "
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "BeginCondition Example"

End Sub
Sub Example_BorderBlockFilename()

    ' This example returns the BorderBlockFilename setting for the current drawing.
    Dim dbPref As AeccDatabasePreferences
    Set dbPref = AeccApplication.ActiveDocument.Preferences

    MsgBox "The current value for BorderBlockFilename is: " & dbPref.BorderBlockFilename, vbInformation, "BorderBlockFilename Example"

End Sub
BorderBottomMargin Example

Sub Example_BorderBottomMargin()

    ' This example returns the BorderBottomMargin setting for the current drawing
    Dim dbPref As AeccDatabasePreferences
    Set dbPref = AeccApplication.ActiveDocument.Preferences

    MsgBox "The current value for BorderBottomMargin is: " & dbPref.BorderBottomMargin
    vbCrLf, "BorderBottomMargin Example"

End Sub
Sub Example_BorderLeftMargin()

    ' This example returns the BorderLeftMargin setting for the current drawing.
    Dim dbPref As AeccDatabasePreferences
    Set dbPref = AeccApplication.ActiveDocument.Preferences

    MsgBox "The current value for BorderLeftMargin is: " & dbPref.BorderLeftMargin
    vbCrLfInformation, "BorderLeftMargin Example"

End Sub
Sub Example_BorderLineWidth()

    ' This example returns the BorderLineWidth setting for the current drawing.
    Dim dbPref As AeccDatabasePreferences
    Set dbPref = AeccApplication.ActiveDocument.Preferences

    MsgBox "The current value for BorderLineWidth is: " & dbPref.BorderLineWidth,
    , "BorderLineWidth Example"

End Sub
BorderPath Example

Sub Example_BorderPath()

    ' This example returns the BorderPath setting.
    Dim prefFiles As AeccPreferencesFiles
    Set prefFiles = AeccApplication.Preferences.Files

    MsgBox "The current value for BorderPath is: " & prefFiles.BorderPath, _
    vbInformation, "BorderPath Example"

End Sub
Sub Example_BorderRightMargin ()

    ' This example returns the BorderRightMargin setting for the current drawing.
    Dim dbPref As AeccDatabasePreferences
    Set dbPref = AeccApplication.ActiveDocument.Preferences

    MsgBox "The current value for BorderRightMargin is: " & dbPref.BorderRightMargin,

End Sub
Sub Example_BorderStyle()

' This example returns the BorderStyle setting for the current drawing.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.Preferences

' Convert the constant to a string.
Dim strStyle As String
If dbPref.BorderStyle = kBorderStyleLine Then
    strStyle = "line."
End If
If dbPref.BorderStyle = kBorderStyleUnscaledBlock Then
    strStyle = "unscaled block."
End If
If dbPref.BorderStyle = kBorderStyleScaledBlock Then
    strStyle = "scaled block."
End If
If dbPref.BorderStyle = kBorderStyleNone Then
    strStyle = "none."
End If

MsgBox "The current value for BorderStyle is " & strStyle, vbInformation, "BorderStyle Example"

End Sub
Sub Example_BorderTopMargin ()

    ' This example returns the BorderTopMargin setting for the current drawing.
    Dim dbPref As AeccDatabasePreferences
    Set dbPref = AeccApplication.ActiveDocument.Preferences

    MsgBox "The current value for BorderTopMargin is: " & dbPref.BorderTopMargin,
         vbInformation, "BorderTopMargin Example"

End Sub
Sub Example_Boudaries()

' This example returns the number of Boundaries in the first ' surface in the collection.
Dim surf As AeccSurface
Dim surfIn As AeccSurfaceInputs
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set surfIn = surf.Inputs

MsgBox "The number of Boundaries in the first Surface is: " & surfIn.Boundaries.Count, vbInformation, "Boundaries Example"

End Sub
Sub Example_Boundary()

' This example returns the first point in the Boundary
' for the first WaterShed in the collection.
Dim surf As AeccSurface
Dim wShed As AeccWaterShed
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set wShed = surf.Outputs.WaterSheds.Item(0)

Dim coords As Variant

coords = wShed.Boundary

MsgBox "The first Boundary point for the first WaterShed is: " & coords(0) & , vbInformation, "Boundary Example"

End Sub
Sub Example_SurfaceInputs_Breaklines()

    ' This example returns the number of Breaklines in the first surface in the collection.
    Dim surf As AeccSurface
    Dim surfIn As AeccSurfaceInputs
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
    Set surfIn = surf.Inputs

    MsgBox "The number of Breaklines in the first Surface is: " & surfIn.Breaklines.Count, vbInformation, "Breaklines Example"

End Sub
Build Example

Examples:

1 Surface

1 WaterSheds

Sub Example_Build_Surfaces()

' This example Builds the first Surface in the collection after adding 20.0 to all elevations.
Dim surf As AeccSurface
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

surf.AddToAllElevations 20#
surf.Build

End Sub

Sub Example_Build_Watersheds()

' This example Builds the Watersheds after increasing the MinDepressionDepth by 10.0
Dim surf As AeccSurface
Dim wSheds As AeccWaterSheds
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set wSheds = surf.Outputs.WaterSheds

Dim minDepth As Double
Dim minArea As Double

minDepth = wSheds.MinDepressionDepth + 10
minArea = wSheds.MinDepressionArea

wSheds.Build MinDepressionDepth, MinDepressionArea, True
End Sub
CCWFlag Example

Examples:

1 **AlignCurve**

1 **ParcelCurve**

---

**Sub** Example_CCWFlag_AlignCurve()

' This example returns the CCWFlag for the first Curve found in the first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Curve entity in the first Alignment."

' Find first Curve in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kCurve Then
        alignMsg = "The CCWFlag for the first Curve in the alignment is: " & alignEnt.CCWFlag
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "CCWFlag Example"

End Sub

---

**Sub** Example_CCWflag_ParcelCurve()

' This example returns the CCWFlag for the first curve in the first Parcel in the collection.
Dim parcel As AeccParcel
Dim parcelEnt As AeccParcelEntity
Set parcel = AeccApplication.ActiveProject.Parcels.Item(0)
Set parcelEnt = parcel.ParcelEntities.Item(0)

Dim parcelMsg As String
parcelMsg = "There is no Curve entity in the first Parcel."

' Find first Curve in the parcel
For Each parcelEnt In parcel.ParcelEntities
    If parcelEnt.Type = kParcelCurve Then
        parcelMsg = "The CCWFlag for the first Curve in the Parcel is: " & parcelEnt.CCWFlag
        Exit For
    End If
Next

MsgBox parcelMsg, vbInformation, "CCWFlag Example"

End Sub
Sub Example_CenterEasting_AlignCurve()

' This example returns the CenterEasting for the first Curve found in the first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Curve entity in the first Alignment."

' Find first Curve in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kCurve Then
        alignMsg = "The CenterEasting for the first Curve in the alignment is: " & Format(alignEnt.CenterEasting, "0.00")
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "CenterEasting Example"
End Sub

Sub Example_CenterEasting_ParcelCurve()

' This example returns the CenterEasting for the first curve in the first Parcel in the collection.

Dim parcel As AeccParcel
Dim parcelEnt As AeccParcelEntity
Set parcel = AeccApplication.ActiveProject.Parcels.Item(0)
Set parcelEnt = parcel.ParcelEntities.Item(0)

Dim parcelMsg As String
parcelMsg = "There is no Curve entity in the first Parcel."

' Find first Curve in the parcel
For Each parcelEnt In parcel.ParcelEntities
    If parcelEnt.Type = kParcelCurve Then
        parcelMsg = "The CenterEasting for the first Curve in the Parcel is: " & Format(parcelEnt.CenterEasting, "0.00")
        Exit For
    End If
Next

MsgBox parcelMsg, vbInformation, "CenterEasting Example"

End Sub
**CenterNorthing Example**

**Examples:**

- [AlignCurve](#)
- [ParcelCurve](#)

```vbnet
Sub Example_CenterNorthing()
    ' This example returns the CenterNorthing for the first Curve found in the first Alignment in the collection.
    Dim align As AeccAlignment
    Dim alignEnts As AeccAlignEntity
    Set align = AeccApplication.ActiveProject.Alignments.Item(0)
    Dim alignMsg As String
    alignMsg = "There is no Curve entity in the first Alignment."
    For Each alignEnt In align.AlignEntities
        If alignEnt.Type = kCurve Then
            alignMsg = "The CenterNorthing for the first Curve in the alignment is: " & Format(alignEnt.CenterNorthing, "0.00")
        Exit For
    End If
    Next
    MsgBox alignMsg, vbInformation, "CenterNorthing Example"
End Sub
```

```vbnet
Sub Example_CenterNorthing_ParcelCurve()
    ' This example returns the CenterNorthing for the first curve in the first Parcel in the collection.
```
```
Dim parcel As AeccParcel
Dim parcelEnt As AeccParcelEntity
Set parcel = AeccApplication.ActiveProject.Parcels.Item(0)
Set parcelEnt = parcel.ParcelEntities.Item(0)

Dim parcelMsg As String
parcelMsg = "There is no Curve entity in the first Parcel."

' Find first Curve in the parcel
For Each parcelEnt In parcel.ParcelEntities
    If parcelEnt.Type = kParcelCurve Then
        parcelMsg = "The CenterNorthing for the first Curve in the Parcel is: " & Format(parcelEnt.CenterNorthing, "0.00")
        Exit For
    End If
Next

MsgBox parcelMsg, vbInformation, "CenterNorthing Example"

End Sub
Sub Example_CentroidEasting()

    ' This example returns the CentroidEasting and the CentroidNorthing
    ' for the first parcel in the collection.
    Dim parcel As AeccParcel
    Set parcel = AeccApplication.ActiveProject.parcels.Item(0)

    MsgBox "The CentroidEasting for the first Parcel in the collection is: " &
    & Format(parcel.CentroidEasting, "0.00") & vbCrLf &
    "The CentroidNorthing for the first Parcel in the collection is: " &
    & Format(parcel.CentroidNorthing, "0.00"), vbInformation, "CentroidEasting Example"

End Sub
Sub Example_CentroidNorthing()

' This example returns the CentroidEasting and the CentroidNorthing
' for the first parcel in the collection.
Dim parcel As AeccParcel
Set parcel = AeccApplication.ActiveProject.parcels.Item(0)

MsgBox "The CentroidEasting for the first Parcel in the collection is: " " 
& Format(parcel.CentroidEasting, "0.00") & vbCrLf & _
"The CentroidNorthing for the first Parcel in the collection is: " _
& Format(parcel.CentroidNorthing, "0.00"), vbInformation, "CentroidNorthing Example"
End Sub
Sub Example_ChordDirection()

' This example returns the ChordDirection for the first Curve found in the
' first Alignment in the collection.
    Dim align As AeccAlignment
    Dim alignEnts As AeccAlignEntity
    Set align = AeccApplication.ActiveProject.Alignments.Item(0)

    Dim alignMsg As String
    alignMsg = "There is no Curve entity in the first Alignment."

    ' Find first Curve in the alignment
    For Each alignEnt In align.AlignEntities
        If alignEnt.Type = kCurve Then
            alignMsg = "The ChordDirection for the first Curve in the alignment is: "
            Exit For
        End If
    Next

    MsgBox alignMsg, vbInformation, "ChordDirection Example"

End Sub
ChordLength Example

Examples:

1 AlignCurve
1 ParcelCurve

Sub Example_ChordLength_AlignCurve()

' This example returns the ChordLength for the first Curve found in the
' first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Curve entity in the first Alignment."

' Find first Curve in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kCurve Then
        alignMsg = "The ChordLength for the first Curve in the alignment is: " _
        & Format(alignEnt.ChordLength, "0.00")
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "ChordLength Example"

End Sub

Sub Example_ChordLength_ParcelCurve()

' This example returns the ChordLength for the first curve in the
' first Parcel in the collection.

Dim parcel As AeccParcel
Dim parcelEnt As AeccParcelEntity
Set parcel = AeccApplication.ActiveProject.Parcels.Item(0)
Set parcelEnt = parcel.ParcelEntities.Item(0)

Dim parcelMsg As String
parcelMsg = "There is no Curve entity in the first Parcel."

' Find first Curve in the parcel
For Each parcelEnt In parcel.ParcelEntities
    If parcelEnt.Type = kParcelCurve Then
        parcelMsg = "The ChordLength for the first Curve in the Parcel is: " 
        & Format(parcelEnt.ChordLength, "0.00")
        Exit For
    End If
Next

MsgBox parcelMsg, vbInformation, "ChordLength Example"

End Sub
Sub Example_CivilDataFilesPath()

' This example returns the CivilDataFilesPath setting.
Dim prefFiles As AeccPreferencesFiles
Set prefFiles = AeccApplication.Preferences.Files

MsgBox "The current value for CivilDataFilesPath is: " & prefFiles.CivilDataFilesPath,
vbInformation, "CivilDataFilesPath Example"

End Sub
ClearOverrides Example

Sub Example_ClearOverrides()

    ' This function clears the overrides for the first PoinGroup in
    ' the collection.
    Dim pntGrp As AeccPointGroup
    Set pntGrp = AeccApplication.ActiveProject.PointGroups.Item(0)

    pntGrp.ClearOverrides

    MsgBox "The overrides have been cleared for PointGroup: " & pntGrp.Group
        vbInformation, "ClearOverrides Example"

End Sub
Sub Example_Code_CrossSectionPointCode()

' This example returns the point code data for the first cross
' section surface in the collection for the first cross section
' in the collection.
Dim aligns As AeccAlignments
Dim align As AeccAlignment
Dim xSect As AeccCrossSection
Dim xSectPCode As AeccCrossSectionPointCode
Set aligns = AeccApplication.ActiveProject.Alignments
Set align = aligns.Item(0)
Set xSect = align.CrossSections.Item(0)
Set xSectPCode = xSect.CrossSectionPointCodes.Item(0)

' Get the alignment name
Dim alignName As String
alignName = align.Name

' Get the cross section station and format it
Dim station As String
station = aligns.DoubleToStaFormat(xSect.station)

MsgBox "The alignment name is: " & alignName & vbCrLf & _
"The first cross section is at station: " & station & vbCrLf & _
"The data for the first point code is: " & vbCrLf & _
vbTab & "Code: " & xSectPCode.Code & vbCrLf & _
vbTab & "Description: " & xSectPCode.Description & vbCrLf & _
vbTab & "Elevation: " & Format(xSectPCode.elevation, "0.00") & vbCrLf & _
vbTab & "Offset: " & Format(xSectPCode.offset, "0.00"), _
vbInformation, "Code Example"

End Sub

Sub Example_Code_DescriptionKey()

' This example returns the Code for the first DescriptionKey
' in the first DescriptionKeyFile in the collection.
Dim dKeyFile As AeccDescriptionKeyFile
Dim dKey As AeccDescriptionKey
Set dKeyFile = AeccApplication.ActiveProject.DescriptionKeyFiles.Item(0)
Set dKey = dKeyFile.Item(0)

MsgBox "The value for DescriptionKey Code is " & dKey.Code, vbInformation

End Sub
Sub Example_Cogo()

' This example returns the text size for Cogo Preferences
' in the current project.
Dim prefPrj As AeccPreferencesProject
Set prefPrj = AeccApplication.ActiveProject.Preferences

MsgBox "The text size for Cogo Preferences in the current Project is: " & pref1 , vbInformation, "Cogo Example"

End Sub
Sub Example_CogoPoints()

    ' This example returns the number of CogoPoints in the current project.
    Dim proj As AeccProject
    Set proj = AeccApplication.ActiveProject

    MsgBox "The number of CogoPoints in the current Project is: " & proj.CogoPoints.Count, vbInformation, "CogoPoints Example"

End Sub
Sub Example_ContourItems()

    ' This example returns the number of ContourItems in the first surface in the collection.
    Dim surf As AeccSurface
    Dim surfIn As AeccSurfaceInputs
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
    Set surfIn = surf.Inputs

    MsgBox "The number of ContourItems in the first Surface is: " & surfIn.ContourItems.Count, _
    vbInformation, "ContourItems Example"

End Sub
Sub Example_ContourStyle()

    ' This example displays the contour style for the selected contour.

    On Error Resume Next

    ' Delete existing SelectionSet
    ThisDrawing.SelectionSets("SSet").Delete

    ' Create the selection set based on a point selection
    ' and filter for Contour objects
    Dim ssetObj As AcadSelectionSet
    Set ssetObj = ThisDrawing.SelectionSets.Add("SSet")

    Dim gpCode(0) As Integer
    Dim dataValue(0) As Variant

    gpCode(0) = 0
    dataValue(0) = "AECC_CONTOUR"

    Dim groupCode As Variant
    Dim dataCode As Variant
    Dim returnPnt As Variant

    groupCode = gpCode
    dataCode = dataValue
    returnPnt = ThisDrawing.Utility.GetPoint(, "Select a contour line: ")
    ssetObj.SelectAtPoint returnPnt, groupCode, dataCode

    Dim objContour As AeccContour
    Set objContour = ssetObj.Item(0)

    MsgBox "The ContourStyle for the selected contour is: " & objContour.ContourStyle.Name, vbInformation, "ContourStyle Example"
End Sub
Sub Example_ContourStyleName()

' This example displays the ContourStyleName for a selected contour

On Error Resume Next

' Delete existing SelectionSet
ThisDrawing.SelectionSets("SSet").Delete

' Create the selection set based on a point selection
' and filter for Contour objects
Dim ssetObj As AcadSelectionSet
Set ssetObj = ThisDrawing.SelectionSets.Add("SSet")

Dim mode As Integer
Dim gpCode(0) As Integer
Dim dataValue(0) As Variant

gpCode(0) = 0
dataValue(0) = "AECC_CONTOUR"

Dim groupCode As Variant
Dim dataCode As Variant
Dim returnPnt As Variant

groupCode = gpCode
dataCode = dataValue
returnPnt = ThisDrawing.Utility.GetPoint(, "Enter a point on a contour line: ")
ssetObj.SelectAtPoint returnPnt, groupCode, dataCode

Dim objContour As AeccContour
Set objContour = ssetObj.Item(0)
MsgBox "The ContourStyleName for the selected contour is " & objContour.ContourStyleName,
     vbInformation, "ContourStyleName Example"
End Sub
Sub Example_ContourStylesPath()

    ' This example returns the ContourStylesPath setting.
    Dim prefFiles As AeccPreferencesFiles
    Set prefFiles = AeccApplication.Preferences.Files

    MsgBox "The current value for ContourStylesPath is: " & prefFiles.ContourStylesPath,
            vbInformation, "ContourStylesPath Example"

End Sub
Sub Example_ConvertToCurrentAreaDisplay()

    ' This example returns the value of ConvertToCurrentAreaDisplay
    ' for the current drawing.
    Dim dbPref As AeccDatabasePreferences
    Set dbPref = AeccApplication.ActiveDocument.preferences

    ' Set the area display units to square yards
    dbPref.AreaDisplayUnit = aecUnitSquareYard

    Dim SourceArea As Double
    Dim TargetArea As Double

    SourceArea = 36# 'Square feet, assumes drawing set to foot units
    TargetArea = dbPref.ConvertToCurrentAreaDisplay(SourceArea)

    MsgBox "The area to convert is 36 square feet." & vbCrLf & _
        "The AreaDisplayUnit is set to square yard." & vbCrLf & _
        "The value for ConvertToCurrentAreaDisplay is " & TargetArea & " square yards.", vbInformation, "ConvertToCurrentAreaDisplay Example"

End Sub
Sub Example_ConvertToCurrentVolumeDisplay()

' This example returns the value of ConvertToCurrentVolumeDisplay
' for the current drawing.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.preferences

' Set the volume display units to cubic yards
dbPref.VolumeDisplayUnit = aecUnitCubicYard

Dim SourceArea As Double
Dim TargetArea As Double

SourceArea = 54# 'Cubic feet, assumes drawing set to foot units
TargetArea = dbPref.ConvertToCurrentVolumeDisplay(SourceArea)

MsgBox "The area to convert is 54 cubic feet." & vbCrLf & _
"The VolumeDisplayUnit is set to cubic yard." & vbCrLf & _
"The value for ConvertToCurrentVolumeDisplay is " & TargetArea & " square yards.", vbInformation, "ConvertToCurrentVolumeDisplay Example"

End Sub
Sub Example_CoordinatePrecision()

' This example returns the CoordinatePrecision setting for the current drawing.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.Preferences

MsgBox "The current value for CoordinatePrecision is: " & dbPref.CoordinatePrecision, vbInformation, "CoordinatePrecision Example"

End Sub
Examples:

- **AeccContour**
- **Boundary**
- **Breakline**
- **CogoPoint**
- **ContourItem**
- **CrossSectionBlock** (Civil Engineering Feature)
- **Edge**
- **ElevationContour**
- **Face**
- **ProfileBlock** (Civil Engineering Feature)

---

```vba
Sub Example_Coordinates_AeccContour()

    ' This example displays the Coordinates for a selected contour
    On Error Resume Next

    ' Delete existing SelectionSet
    ThisDrawing.SelectionSets("SSet").Delete

    ' Create the selection set based on a point selection
    ' and filter for Contour objects
    Dim ssetObj As AcadSelectionSet
    Set ssetObj = ThisDrawing.SelectionSets.Add("SSet")
```

---
Sub Example_Coordinates_Boundary()

    ' This example returns the first point in the Coordinates
    ' for the first Boundary in the collection.
    Dim surf As AeccSurface
    Dim bound As AeccBoundary
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
    Set bound = surf.Inputs.Boundaries.Item(0)

    Dim coords As Variant

    coords = objContour.Coordinates

    MsgBox "The first point in Coordinates is: " & coords(0) & ", " & coords(1) & 
        vbInformation, "Coordinates Example"

End Sub

Sub Example_Coordinates_Boundary()

    ' This example returns the first point in the Coordinates
    ' for the first Boundary in the collection.
    Dim surf As AeccSurface
    Dim bound As AeccBoundary
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
    Set bound = surf.Inputs.Boundaries.Item(0)

    Dim coords As Variant

   (coords) = 0
    dataValue(0) = "AECC_CONTOUR"

    Dim groupCode As Variant
    Dim dataCode As Variant
    Dim returnPnt As Variant

    groupCode = gpCode
    dataCode = dataValue
    returnPnt = ThisDrawing.Utility.GetPoint(, "Enter a point on a contour line: ")
    ssetObj.SelectAtPoint returnPnt, groupCode, dataCode

    Dim objContour As AeccContour
    Set objContour = ssetObj.Item(0)

    Dim coords As Variant

    coords = objContour.Coordinates

    MsgBox "The first point in Coordinates is: " & coords(0) & ", " & coords(1) & 
        vbInformation, "Coordinates Example"
coords = bound.Coordinates

MsgBox "The first Coordinate for the Boundary is: " & coords(0) & ", " & coords(1) & ", " & coords(2) & ", vbInformation, "Coordinates Example"

End Sub

Sub Example_Coordinates_BreakLine()

' This example returns the first point in the Coordinates
' for the first BreakLine in the collection.
Dim surf As AeccSurface
Dim brkLine As AeccBreakLine
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set brkLine = surf.Inputs.BreakLines.Item(0)

Dim coords As Variant

coords = brkLine.Coordinates

MsgBox "The first Coordinate for the BreakLine is: " & coords(0) & ", " & coords(1) & ", " & coords(2) & ", vbInformation, "Coordinates Example"

End Sub

Sub Example_Coordinates_CogoPoint()

' This example returns the Coordinate for the first CogoPoint in the collection
Dim cogoPnt As AeccCogoPoint
Set cogoPnt = AeccApplication.ActiveProject.CogoPoints.Item(0)

Dim coords As Variant

coords = cogoPnt.Coordinates

MsgBox "The Coordinates for the first CogoPoint in the collection is: " & coords(0) & ", " & coords(1) & ", " & coords(2) & ", vbInformation, "Coordinates Example"
Sub Example_Coordinates_ContourItem()

' This example returns the first point in the Coordinates
' for the first ContourItem in the collection.
Dim surf As AeccSurface
Dim cont As AeccContourItem
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set cont = Surf.Inputs.ContourItems.Item(0)

Dim coords As Variant
coords = cont.Coordinates

MsgBox "The first Coordinates for the ContourItem is: " & coords(0) & ", " & coords(1) & ", " & coords(2), vbInformation, "Coordinates Example"

End Sub

Sub Example_Coordinates_CrossSectionBlock()

' This example returns the coordinates for the
' first alignment cross section in the collection.
Dim alignXsects As AeccCrossSectionBlocks
Dim alignXsect As AeccCrossSectionBlock
Set alignXsects = AeccApplication.ActiveDocument.CrossSectionBlocks
Set alignXsect = alignXsects.Item(0)

' Get the station for the first alignment cross section in the collection
Dim station As String
station = alignXsect.station

' Get the coordinates for the first alignment cross section in the collection
Dim coords As Variant
coords = alignXsect.Coordinates

Sub Example_Coordinates_Edge()

    ' This example returns the coordinates for the first
    ' Edge in the collection.
    Dim surf As AeccSurface
    Dim edge As AeccEdge
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
    Set edge = Surf.Outputs.Edges.Item(0)

    Dim coords As Variant
    coords = edge.Coordinates

    MsgBox "The Coordinates for the first Edge are:" & vbCrLf & 
        "FromNorthing: " & coords(0) & vbCrLf & 
        "FromEasting: " & coords(1) & vbCrLf & 
        "FromElevation: " & coords(2) & vbCrLf & 
        "ToNorthing: " & coords(3) & vbCrLf & 
        "ToEasting: " & coords(4) & vbCrLf & 
        "ToElevation: " & coords(5), vbInformation, "Coordinates Example"

End Sub

Sub Example_Coordinates_ElevationContour()

    ' This example returns the first point in the coordinates for the
    ' first ElevationContour in the collection at 100.0
    Dim surf As AeccSurface
    Dim elevContours As AeccElevationContours
    Dim elevContour As AeccElevationContour
    Dim coords As Variant
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set elevContours = Surf.Outputs.ElevationContours

'Create a collection of ElevationContours at 100.0
elevContours.Elevation = 100#

Set elevContour = elevContours.Item(0)

coords = elevContour.Coordinates

MsgBox "The first point in Coordinates is: " & coords(0) & ", " & coords(1) & 

End Sub

Sub Example_Coordinates_Face()

' This example returns the Coordinates
' for the first Face in the collection.
Dim surf As AeccSurface
Dim face As AeccFace
Set Surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set face = surf.Outputs.faces.Item(0)

Dim coord As Variant

coord = face.Coordinates

MsgBox "The points for the first face are: " & vbCrLf & _
coord(0) & ", " & coord(1) & ", " & coord(2) & vbCrLf & _
coord(3) & ", " & coord(4) & ", " & coord(5) & vbCrLf & _
coord(6) & ", " & coord(7) & ", " & coord(8), vbInformation, "Coordinates I

End Sub

Sub Example_Coordinates_ProfileBlock()
This example returns the Coordinate for the first ProfileBlock in the collection.

Dim alignProf As AeccProfileBlock
Set alignProf = AeccApplication.ActiveDocument.ProfileBlocks.Item(0)

Dim coords As Variant
coords = alignProf.Coordinates

MsgBox "The Coordinates for the first Alignment Profile are:" & vbCrLf &
"X Value: " & Format(coords(0), "0.00") & vbCrLf & 
"Y Value: " & Format(coords(1), "0.00"), vbInformation, "Coordinates Example"

End Sub
CoordinateZone Example

Examples:

- **DatabasePreferences**
- **DEMFile**

```vba
Sub Example_CoordinateZone_DatabasePreferences()
    ' This example returns the CoordinateZone setting for the current drawing.
    Dim dbPref As AeccDatabasePreferences
    Set dbPref = AeccApplication.ActiveDocument.Preferences

    MsgBox "The current value for CoordinateZone is: " & dbPref.CoordinateZone,
    vbInformation, "CoordinateZone Example"
End Sub
```

```
Sub Example_CoordinateZone_DEMFile()
    ' This example returns the CoordinateZone for the first DEMFile in the first surface in the collection.
    Dim surf As AeccSurface
    Dim DEMFile As AeccDEMFile
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
    Set DEMFile = surf.Inputs.DEMFiles.Item(0)

    MsgBox "The CoordinateZone for the first DEMFile is: " & DEMFile.CoordinateZone,
    vbInformation, "CoordinateZone Example"
End Sub
```
Copy Example

Examples

1 Prototypes

1 Surfaces

---

Sub Example_Copy_Prototypes()

' This example copies a the first prototype in the collection
to a "Copy of " the first prototype..
Dim prots As AeccPrototypes
Dim prot As AeccPrototype
Dim protcopy As AeccPrototype
Set prots = AeccApplication.Prototypes
Set prot = prots.Item(0)

Dim source As String
Dim target As String

' Get the source and target prototype names
source = prot.Name
target = "Copy of " & source

' Make copy of prototype
Set protcopy = prots.Copy(source, target)

MsgBox "The copied prototype is named: " & protcopy.Name, vbInformation,

End Sub

---

Sub Example_Copy_Surfaces()

' This example copies a surface.
Dim surfs As AeccSurfaces
Dim surf As AeccSurface
Dim surfCopy As AeccSurface
Set surfs = AeccApplication.ActiveProject.Surfaces
Set surf = surfs.Item(0)

Dim source As String

' Get name of surface to copy
source = surf.Name

Set surfCopy = surfs.Copy(source)

MsgBox "The copied surface is named: " & surfCopy.Name, vbInformation, "(

End Sub
Sub Example_Count()

' Use count to retrieve the number of entities in a collection.
' You might use this value in a loop structure to iterate through the collection.
MsgBox "There are " & AeccApplication.Projects.count & " project(s) at the project path.", vbInformation, "Count Example"

Dim objCount As Integer
objCount = AeccApplication.Projects.count

Dim I As Integer
Dim proj As AeccProject
For I = 0 To objCount - 1
    Set proj = AeccApplication.Projects.Item(I)
    MsgBox "The projects collection includes: " & proj.Name, vbInformation, "Count Example"
Next

End Sub
Course Example

Sub Example_Course()

    ' This example returns the Course for the first entity in the
    ' first Parcel in the collection.
    Dim parcel As AeccParcel
    Dim parcelEnt As AeccParcelEntity
    Set parcel = AeccApplication.ActiveProject.Parcels.Item(0)
    Set parcelEnt = parcel.ParcelEntities.Item(0)

    MsgBox "The Course of the first entity in the Parcel is: " & parcelEnt.Course, ,
        vbInformation, "Course Example"

End Sub
CourseIn Example

Sub Example_CourseIn()

' This example returns the CourseIn for the first curve in the
' first Parcel in the collection.
Dim parcel As AeccParcel
Dim parcelEnt As AeccParcelEntity
Set parcel = AeccApplication.ActiveProject.Parcels.Item(0)
Set parcelEnt = parcel.ParcelEntities.Item(0)

Dim parcelMsg As String
parcelMsg = "There is no Curve entity in the first Parcel."

' Find first Curve in the parcel
For Each parcelEnt In parcel.ParcelEntities
    If parcelEnt.Type = kParcelCurve Then
        parcelMsg = "The CourseIn for the first Curve in the Parcel is: " _
        & parcelEnt.CourseIn & vbCrLf _
        & "The CourseOut for the first Curve in the Parcel is: " _
        & parcelEnt.CourseOut
        Exit For
    End If
    Next

MsgBox parcelMsg, vbInformation, "CourseIn Example"

End Sub
CourseOut Example

Sub Example_CourseOut()

' This example returns the CourseOut for the first curve in the
' first Parcel in the collection.
Dim parcel As AeccParcel
Dim parcelEnt As AeccParcelEntity
Set parcel = AeccApplication.ActiveProject.Parcels.Item(0)
Set parcelEnt = parcel.ParcelEntities.Item(0)

Dim parcelMsg As String
parcelMsg = "There is no Curve entity in the first Parcel."

' Find first Curve in the parcel
For Each parcelEnt In parcel.ParcelEntities
    If parcelEnt.Type = kParcelCurve Then
        parcelMsg = "The CourseIn for the first Curve in the Parcel is: " _
        & parcelEnt.CourseIn & vbCrLf _
        & "The CourseOut for the first Curve in the Parcel is: " _
        & parcelEnt.CourseOut
        Exit For
    End If
Next

MsgBox parcelMsg, vbInformation, "CourseOut Example"

End Sub
Sub Example_CrossSection()

' This example returns the datum layer for the cross section preferences in the current project.
Dim prefPrj As AeccPreferencesProject
Set prefPrj = AeccApplication.ActiveProject.Preferences

MsgBox "The datum layer for cross section preferences in the current project is:"
    & prefPrj.CrossSection.GetString(kDatumLayer), vbInformation, "Profile Example"

End Sub
Sub Example_CrossSectionBlocks()

' This example returns the count of cross section blocks in
' the current document.
Dim doc As AeccDocument
Set doc = AeccApplication.ActiveDocument

MsgBox "The number of cross section blocks in the current document is " & _
       doc.CrossSectionBlocks.Count, vbInformation, "CrossSectionBlocks Exampl"
Sub Example_CrossSectionPointCodes()

    ' This example returns the count of point codes for the 
    ' first cross section in the collection.
    Dim align As AeccAlignment
    Dim xSect As AeccCrossSection
    Set align = AeccApplication.ActiveProject.Alignments.Item(0)
    Set xSect = align.CrossSections.Item(0)

    MsgBox "The number of point codes in the first cross section is: " _
        & xSect.CrossSectionPointCodes.Count, vbInformation, "CrossSectionPoint

End Sub
Sub Example_CrossSections()

    ' This example returns the number of cross sections for the
    ' first alignment in the collection.
    Dim align As AeccAlignment
    Set align = AeccApplication.ActiveProject.Alignments.Item(0)

    MsgBox "The number of cross sections in first alignment is: " _
        & align.CrossSections.Count, vbInformation, "CrossSections Example"

End Sub
Sub Example_CrossSectionSurfaces()

' This example returns the count of surfaces for the
' first cross section in the collection.
Dim align As AeccAlignment
Dim xSect As AeccCrossSection
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set xSect = align.CrossSections.Item(0)

MsgBox "The number of surfaces in the first cross section is: " _
& xSect.CrossSectionSurfaces.Count, vbInformation, "CrossSectionSurfaces";

End Sub
Sub Example_CurrentAlignment()

' This example returns the CurrentAlignment Value.
Dim aligns As AeccAlignments
Set aligns = AeccApplication.ActiveProject.Alignments

MsgBox "The CurrentAlignment for alignments is " & aligns.CurrentAlignment,
vbInformation, "CurrentAlignment Example"

End Sub
CurrentSurface Example

Sub Example_CurrentSurface()

    ' This example returns the CurrentSurface setting.
    Dim surfs As AeccSurfaces
    Set surfs = AeccApplication.ActiveProject.Surfaces

    MsgBox "The CurrentSurface is: " & surfs.CurrentSurface, vbInformation, "CurrentSurface Example"

End Sub
CurveCode Example

Sub Example_CurveCode()

' This example returns the data for the first superelevation in the
' superelevation collection for the first alignment in the collection.
Dim aligns As AeccAlignments
Dim align As AeccAlignment
Dim sElev As AeccSuperelevation
Set aligns = AeccApplication.ActiveProject.Alignments
Set align = aligns.Item(0)
Set sElev = align.Superelevations.Item(0)

' Get the alignment name
Dim alignName As String
alignName = align.Name

' Get the superelevation station and format it
Dim station As String
station = aligns.DoubleToStaFormat(sElev.station)

' Get the superelevation curve code
Dim crvCode As String
Select Case sElev.curveCode
Case kSERightHandCurve
    crvCode = "Right Hand Curve"
Case kSELeftHandCurve
    crvCode = "Left Hand Curve"
End Select

' Get the superelevation code
Dim supCode As String
Select Case sElev.SuperelevationCode
Case kSEFullCrown
    supCode = "Full Crown"
Case kSEHalfCrown
supCode = "Half Crown"

Case kSECrownRemoved
    supCode = "Crown Removed"
Case kSEFullSuperelevations
    supCode = "Full Superelevation"
Case kSEReverseCurve
    supCode = "Reverse Curve"
Case kSECompoundCurve
    supCode = "Compound Curve"
End Select

MsgBox "The alignment name is: " & alignName & vbCrLf & 
    "The data for the first superelevation is: " & vbCrLf & _
    vbCrLf & vbTab & "Station: " & station & vbCrLf & _
    vbCrLf & vbTab & "Curve Code: " & crvCode & vbCrLf & _
    vbCrLf & vbTab & "Superelevation Code: " & supCode, _
    vbInformation, "CurveCode Example"

End Sub
Sub Example_CurveLabelIncrement()

' This example returns the CurveLabelIncrement for the first ProfileBlock
' in the collection
Dim alignProf As AeccProfileBlock
Set alignProf = AeccApplication.ActiveDocument.ProfileBlocks.Item(0)

MsgBox "The CurveLabelIncrement for the first ProfileBlock in the collection
& alignProf.CurveLabelIncrement, vbInformation, "CurveLabelIncrement I"

End Sub
CurveLength Example

Sub Example_CurveLength()

' This example returns the CurveLength value for the second PVI in the
' first finished ground of the first alignment in the collection.
Dim align As AeccAlignment
Dim FGProf As AeccFGProfile
Dim PVI As AeccPVI
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set FGProf = align.FGProfiles.Item(0)
Set PVI = FGProf.PVIs.Item(1)

MsgBox "The CurveLength of the second PVI is: " & Format(PVI.CurveLength, "0.000"), vbInformation, "CurveLength Examț

End Sub
DatabaseScale Example

Sub Example_DatabaseScale()
    ' This example returns the HorizontalScale setting for the current drawing.
    Dim dbPref As AeccDatabasePreferences
    Set dbPref = AeccApplication.ActiveDocument.Preferences
    MsgBox "The current value for DatabaseScale is: " & dbPref.DatabaseScale, _
        vbInformation, "DatabaseScale Example"
End Sub
Sub Example_DataPath()

' This example returns the DataPath setting.
Dim prefFiles As AeccPreferencesFiles
Set prefFiles = AeccApplication.Preferences.Files

MsgBox "The current value for DataPath is: " & prefFiles.DataPath, _
    vbInformation, "DataPath Example"

End Sub
DatumElevation Example

Examples:

1 CrossSectionBlock (Civil Engineering Feature)

1 ProfileBlock (Civil Engineering Feature)

Sub Example_DatumElevation_CrossSectionBlock()

    ,

    This example returns the datum elevation for the
    ,

    first alignment cross section in the collection.
    Dim
    alignXsects As AeccCrossSectionBlocks
    Dim
    alignXSect As AeccCrossSectionBlock
    Set
    alignXsects = AeccApplication.ActiveDocument.CrossSectionBlocks
    Set
    alignXSect = alignXsects.Item(0)

    'Get
    the station for the first alignment cross section in the collection
    Dim
    station As String
    station
    = alignXSect.station

    MsgBox
    "The datum elevation for the alignment cross section at station " & station & " is: " & _
    Format(alignXSect.DatumElevation, "0.00"), vbInformation, "DatumElevation Example"
This example returns the DatumElevation for the first ProfileBlock in the collection.

Dim alignProf As AeccProfileBlock
Set alignProf = AeccApplication.ActiveDocument.ProfileBlocks.Item(0)

MsgBox "The DatumElevation for the first ProfileBlock in the collection is: " & alignProf.DatumElevation, vbInformation, "DatumElevation Example"
Delete Example

Examples:

- Alignments
- Boundaries
- BreakLines
- CogoPoints
- ContourItems
- DEMFiles
- DescriptionKeyFile
- DescriptionKeyFiles
- EGProfiles (Civil Engineering Feature)
- FGProfiles (Civil Engineering Feature)
- Parcels
- PointFiles
- PointGroupNames
- PointGroups
- Projects
- Prototypes
- PVIs (Civil Engineering Feature)
- StationEquations
Sub Example_Delete_Alignments()

' This example deletes an Alignment from the Alignments Collection
Dim aligns As AeccAlignments
Set aligns = AeccApplication.ActiveProject.Alignments
' Show the number of alignments in the project
MsgBox "The initial Count of Alignments is: " & aligns.Count, vbInformation,

' Add an Alignment named New Alignment and starting at Station 2000
aligns.Add "New Alignment", 2000

' Show the number of alignments in the project
MsgBox "The Count of Alignments after Add is: " & aligns.Count, vbInformation,

' Deletes the Alignment named New Alignment
aligns.Delete "New Alignment"

' Show the number of alignments in the project
MsgBox "The Count of Alignments after Delete is: " & aligns.Count, vbInformation,

End Sub

Sub Example_Delete_Boundaries()

' This example starts by displays the initial count of the Boundaries
' for the first surface in the collection. A new visible Boundary is added
' and the count is redisplayed. Finally, the new Boundary is deleted. The count is
' displayed again, showing the deletion.
Dim surf As AeccSurface
Dim bounds As AeccBoundaries
Dim Bound As AeccBoundary
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set bounds = surf.Inputs.Boundaries
MsgBox "The Boundaries Count is: " & bounds.count, vbInformation, "Delete

' Initialize variables
Dim pnts(0 To 14) As Double
Dim count As Integer
Dim index As Integer
Dim pnt As Variant

index = 0

' Add a new Boundary based on five selected points
For count = 1 To 5
    pnt = ThisDrawing.Utility.GetPoint(, "Select point" + Str(count) + " of Boundary:")
pnts(index) = pnt(0): pnts(index + 1) = pnt(1): pnts(index + 2) = pnt(2)
    index = index + 3
Next count

' Add a new Boundary
Set Bound = bounds.Add(kBoundaryTypeShow, False, pnts, "NewBoundary")

MsgBox "The BreakLines Count after the add is: " & bounds.count, vbInformation

' Delete the new Boundary
bounds.Delete Bound.Id

MsgBox "The Boundaries Count after the delete is " & bounds.count, vbInformation

End Sub

Sub Example_Delete_BreakLines()

' This example starts by displays the initial count of the BreakLines
' for the first surface in the collection. A new BreakLines is added
' and the count is redisplayed. Finally, the new file is deleted. The count is
' displayed again, showing the deletion.
Dim surf As AeccSurface
Dim brkLines As AeccBreakLines
Module Dim brkLine As AeccBreakLine
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
    Set brkLines = surf.Inputs.BreakLines

    MsgBox "The BreakLines Count is: " & brkLines.count, vbInformation, "Delete Example"

    ' Initialize variables
    Dim pnts(0 To 14) As Double
    Dim count As Integer
    Dim index As Integer
    Dim pnt As Variant

    index = 0

    ' Add a new BreakLine based on five selected points
    For count = 1 To 5
        pnt = ThisDrawing.Utility.GetPoint(, "Select point" + Str(count) + " of BreakLine: ")
        pnts(index) = pnt(0): pnts(index + 1) = pnt(1): pnts(index + 2) = pnt(2)
        index = index + 3
    Next count

    Set brkLine = brkLines.Add(pnts, "NewBreakLine")

    MsgBox "The BreakLines Count after the add is: " & brkLines.count, vbInformation

    ' Delete the new BreakLines
    brkLines.Delete brkLine.Id
    MsgBox "The BreakLines Count after the delete is " & brkLines.count, vbInformation

End Sub

Sub Example_Delete_CogoPoints()

    ' This example displays the used point numbers, deletes a point number
    ' and the displays the used point numbers again to confirm the deletion.
    Dim cogoPnts As AeccCogoPoints
    Set cogoPnts = AeccApplication.ActiveProject.CogoPoints
MsgBox "Used point numbers are " & cogoPnts.UsedPointNumbers, _
vbInformation, "Delete Example"

' Get a point number
Dim pntNum As Long
pntNum = ThisDrawing.Utility.GetReal("Enter a valid point number : ")
pntNum = Fix(pntNum)

' Delete the CogoPoint
cogoPnts.Delete pntNum

MsgBox "Used point numbers are " & cogoPnts.UsedPointNumbers, _
vbInformation, "Delete Example"
End Sub

Sub Example_Delete_CoContourItems()

' This example Deletes the ContourItem with ID equal to one (1)
' from the first collection of ContourItems.
Dim surf As AeccSurface
Dim conts As AeccContourItems
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set conts = surf.Inputs.ContourItems

' Display the number of ContourItems in the collection before the delete
MsgBox "The number of ContourItems in the collection is: " & conts.Count, _
vbInformation, "Delete Example"

' Delete ContourItem with ID equal to one.
conts.Delete 1

' Display the number of ContourItems in the collection after the delete
MsgBox "The number of ContourItems in the collection is: " & conts.Count, _
vbInformation, "Delete Example"
Sub Example_Delete_DEMFiles()

' This example starts by displaying the initial count of DEMFiles
' for the first surface. A new DEMfile is added and the count is redisplays.
' Finally, the new file is deleted. The count is displayed again, showing
' the deletion.
Dim surf As AeccSurface
Dim DEMFiles As AeccDEMFiles
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set DEMFiles = surf.Inputs.DEMFiles

MsgBox "The DEMFiles Count is: " & DEMFiles.Count, vbInformation, "Delete Example"

' Get the DEMFile name and format the prompt
Dim DEMName As String
Dim prompt As String

prompt = "Enter the name of the DEMFile for surface " & surf.Name & ": "
DEMName = ThisDrawing.Utility.GetString(False, prompt)

' Add a new DEMFile
DEMFiles.Add DEMName

MsgBox "The DEMFiles Count after the add is: " & DEMFiles.Count, vbInformation

' Delete the new DEMFile
DEMFiles.Delete DEMName

MsgBox "The DEMFiles Count after the delete is " & DEMFiles.Count, vbInformation

End Sub

Sub Example_Delete_DescriptionKeyFile()

' This example starts by displaying the initial count of DescriptionKeys

' in the DEFAULT DescriptionKey file. A new DescriptionKey is added
' and the count is redispals. Finally, the new key is deleted. The
' count is displayed again, showing the deletion.
Sub Example_Delete_DescriptionKeyFiles()

' This example starts by displays the initial count of DescriptionKeyFiles
' on the system. A new DescriptionKeyFile is added and the count is redispals
' Finally, the new file is deleted. The count is displayed again, showing
' the deletion.
Dim dKeyFiles As AeccDescriptionKeyFiles
Set dKeyFiles = AeccApplication.ActiveProject.DescriptionKeyFiles

MsgBox "The DescriptionKeyFiles Count is " & dKeyFiles.count, vbInformation

' Add a new DescriptionKeyFile
dKeyFiles.Add "NewFile"

MsgBox "The DescriptionKeyFiles Count is " & dKeyFiles.count, vbInformation

' Delete the new DescriptionKeyFile
dKeyFiles.Delete "NewFile"

MsgBox "The DescriptionKeyFiles Count is " & dKeyFiles.count, vbInformation

End Sub

Sub Example_Delete_EGProfiles()

' This example deletes the first existing ground profile
' in the first alignment in the collection.
Dim align As AeccAlignment
Dim EGProf As AeccEGProfile
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set EGProf = align.EGProfiles.Item(0)

MsgBox "The number of existing ground profiles in the first alignment is: " _
& align.EGProfiles.Count, vbInformation, "Delete Example"

Dim surfName As String
Dim offset As Integer
surfName = EGProf.SurfaceName
offset = EGProf.Type

' Delete the profile
align.EGProfiles.Delete offset, surfName

MsgBox "The number of existing ground profiles in the first alignment is: " _
& align.EGProfiles.Count, vbInformation, "Delete Example"

End Sub

Sub Example_Delete_FGProfiles()

' This example deletes the first finished ground profile
' in the first alignment in the collection.
Dim align As AeccAlignment
Dim FGProf As AeccFGProfile

Dim surfName As String
Dim offset As Integer
surfName = FGProf.SurfaceName
offset = FGProf.Type

' Delete the profile
align.FGProfiles.Delete offset, surfName

MsgBox "The number of existing finished ground profiles in the first alignment is: " _
& align.FGProfiles.Count, vbInformation, "Delete Example"

End Sub
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set FGProf = align.FGProfiles.Item(0)

MsgBox "The number of finished ground profiles in the first alignment is: " 
& align.fGProfiles.Count, vbInformation, "Delete Example"

Dim offset As Integer
offset = FGProf.Type

' Delete the profile
align.FGProfiles.Delete offset

MsgBox "The number of finished ground profiles in the first alignment is: " 
& align.FGProfiles.Count, vbInformation, "Delete Example"

End Sub

Sub Example_Delete_Parcels()

' This example starts by displaying the initial count of Parcels
' A new Parcel is added and the count is redisplayed. Finally,
' the new Parcel is deleted. The count is displayed again, showing
' the deletion.

Dim parcels As AeccParcels
Set parcels = AeccApplication.ActiveProject.Parcels

MsgBox "The Parcel Count is: " & parcels.Count, vbInformation, "Add Example"

' Add a new Parcel
parcels.Add "NewParcel"

MsgBox "The Parcel Count is: " & parcels.Count, vbInformation, "Add Example"

' Delete the new Parcel
parcels.Delete "NewParcel"
MsgBox "The Parcel Count is: " & parcels.Count, vbInformation, "Add Example"

End Sub

Sub Example_Delete_PointFiles()

' This example starts by displays the initial count of PointFiles
' on the system. A new Pointfile is added and the count is redisplays.
' Finally, the new file is deleted. The count is displayed again, showing
' the deletion.
Dim surf As AeccSurface
Dim pntFiles As AeccPointFiles
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set pntFiles = surf.Inputs.PointFiles

MsgBox "The PointFiles Count is: " & pntFiles.Count, vbInformation, "Delete Example"

' Add a new PointFile
pntFiles.Add "NewPointFile"

MsgBox "The PointFiles Count after the add is: " & pntFiles.Count, vbInformation, "Delete Example"

' Delete the new PointFile
pntFiles.Delete "NewPointFile"

MsgBox "The PointFiles Count after the delete is " & pntFiles.Count, vbInformation, "Delete Example"

End Sub

Sub Example_Delete_PointGroupNames()

' This example starts by displays the initial count of the PointGroupNames
' for the first surface in the collection. A new PointGroupName is added
' and the count is redisplays. Finally, the new file is deleted. The count is
' displayed again, showing the deletion.
Dim surf As AeccSurface
Dim pntGrpNames As AeccPointGroupNames
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set pntGrpNames = surf.Inputs.PointGroupNames

MsgBox "The PointGroupNames Count is: " & pntGrpNames.Count, vbInformation, "Delete Example"

' Add a new PointGroupName
pntGrpNames.Add "NewPointGroupName"

MsgBox "The PointGroupNames Count after the add is: " & pntGrpNames.Count, vbInformation, "Delete Example"

' Delete the new PointGroupName
pntGrpNames.Delete "NewPointGroupName"

MsgBox "The PointGroupNames Count after the delete is " & pntGrpNames.Count, vbInformation, "Delete Example"

End Sub

Sub Example_Delete_Projects()

' This example creates a new project named "New Project" and then deletes the project. A project count is displayed during the process.
Dim projs As AeccProjects
Set projs = AeccApplication.Projects

' Show the number of projects
MsgBox "The initial Count of Projects is: " & projs.Count, vbInformation, "Delete Example"
projs.Add "New Project", "Default (Feet)"

' Show the number of projects after the Add
MsgBox "The Count of Projects after Add is: " & projs.Count, vbInformation, "Delete Example"
projs.Delete "New Project"

' Show the number of projects after the Delete
MsgBox "The Count of Projects after Delete is: " & projs.Count, vbInformation, "Delete Example"
Sub Example_Delete_Prototypes()

' This example deletes the first prototype in the collection.
Dim prots As AeccPrototypes
Dim prot As AeccPrototype
Set prots = AeccApplication.Prototypes
Set prot = prots.Item(0)

' Get the source prototype names
Dim source As String
source = prot.Name

' Delete the prototype
prots.Delete source

MsgBox "The deleted prototype was named: " & source, vbInformation, "Delete Example"

End Sub

Sub Example_Delete_PointGroups()

' This example adds and deletes a PointGroup from the PoinGroups Collection.
Dim pntGrps As AeccPointGroups
Set pntGrps = AeccApplication.ActiveProject.PointGroups

' Show the number of PointGroups in the project
MsgBox "The initial Count of PointGroups is: " & pntGrps.Count, vbInformation

' Add PointGroup
pntGrps.Add "New Group", "1-100"

' Show the number of PointGroups in the project after the Add
MsgBox "The Count of PointGroups is: " & pntGrps.Count, vbInformation

' Deletes the PointGroup named "New Group"
pntGrps.Delete "New Group"
' Show the number of PointGroups in the project after the Delete
MsgBox "The Count of PointGroups is: " & pntGrps.Count, vbInformation, "D

End Sub

Sub Example_Delete_PVIs()

' This example deletes the second PVI in the first finished
' ground profile of the first alignment in the collection.
Dim align As AeccAlignment
Dim FGProf As AeccFGProfile
Dim PVI As AeccPVI
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set FGProf = align.FGProfiles.Item(0)
Set PVI = FGProf.PVIs.Item(1)

MsgBox "The number of PVIs in the finished ground profile is: " _
    & FGProf.PVIs.Count, vbInformation, "Delete Example"

' Get the station and the elevation
Dim station As Double
station = PVI.station

' Delete the second PVI
FGProf.PVIs.Delete station

MsgBox "The number of PVIs in the finished ground profile is: " _
    & FGProf.PVIs.Count, vbInformation, "Delete Example"

End Sub

Sub Example_Delete_StationEquations()

' This example deletes the first StationEquations from
' the first alignment in the collection.
Dim aligns As AeccAlignments
Dim align As AeccAlignment
Set aligns = AeccApplication.ActiveProject.Alignments

' Get the first alignment
Set align = aligns.Item(0)

' Set the first alignment current.
aligns.CurrentAlignment = align.Name

Dim staEqus As AeccStationEquations
Set staEqus = align.StationEquations

' Show the StationEquation count for the first alignment before the delete
MsgBox "The StationEquation count for the first alignment is: " & staEqus.count, vbInformation, "Delete Example"

' Delete the first StationEquation
staEqus.Delete(0)

' Show the StationEquation count for the first alignment after the delete
MsgBox "The StationEquation count for the first alignment is: " & staEqus.count, vbInformation, "Delete Example"

End Sub

Sub Example_Delete_Surfaces()

' This example deletes the first surface from the collection.
Dim surfs As AeccSurfaces
Dim surf As AeccSurface
Set surfs = AeccApplication.ActiveProject.Surfaces
Set surf = surfs.Item(0)

' Get the name of the first surface in the collection
Dim surfName As String
surfName = surf.Name
' Show the Surfaces count before the delete
MsgBox "The Surface count is: " & surfs.Count, vbInformation, "Delete Exam

' Delete the first StationEquation
surfs.Delete (surfName)

' Show the Surfaces count after the delete
MsgBox "The Surfaces count is: " & surfs.Count, vbInformation, "Delete Exa

End Sub
Delta Example

Examples:

- AlignCurve
- AlignSpiral
- ParcelCurve

Sub Example_Delta_AlignCurve()

' This example returns the Delta for the first Curve found in the first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Curve entity in the first Alignment."

' Find first Curve in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kCurve Then
        alignMsg = "The Delta for the first Curve in the alignment is: " & Format(alignEnt.Delta, "0.00")
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "Delta Example"

End Sub

Sub Example_Delta_AlignSpiral()

' This example returns the Delta for the first Spiral found in the
' first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Spiral entity in the first Alignment."

' Find first Spiral in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kSpiral Then
        alignMsg = "The Delta for the first Spiral in the alignment is: " 
                   & Format(alignEnt.Delta, "0.00")
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "Delta Example"
End Sub

Sub Example_Delta_ParcelCurve()

    ' This example returns the Delta for the first curve in the
    ' first Parcel in the collection.
    Dim parcel As AeccParcel
    Dim parcelEnt As AeccParcelEntity
    Set parcel = AeccApplication.ActiveProject.Parcels.Item(0)
    Set parcelEnt = parcel.ParcelEntities.Item(0)

    Dim parcelMsg As String
    parcelMsg = "There is no Curve entity in the first Parcel."

    ' Find first Curve in the parcel
    For Each parcelEnt In parcel.ParcelEntities
        If parcelEnt.Type = kParcelCurve Then
            parcelMsg = "The Delta for the first Curve in the Parcel is: " 

```
& Format(parcelEnt.Delta, "0.00")
Exit For
End If
Next

MsgBox parcelMsg, vbInformation, "Delta Example"

End Sub
Sub Example_DEMFiles()

' This example returns the number of DEMFiles in the first
' surface in the collection.
Dim surf As AeccSurface
Dim surfIn As AeccSurfaceInputs
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set surfIn = surf.Inputs

MsgBox "The number of DEMFiles in the first Surface is: " & surfIn.DEMFiles.Count, vbInformation, "DEMFiles Example"

End Sub
Description Example

Examples:

- **AeccPoint**
- **Alignment**
- **Boundary**
- **BreakLine**
- **CrossSectionPointCode** (Civil Engineering Feature)
- **PointGroup**
- **Project**
- **Prototype**
- **Surface**

---

Sub Example_Description_AeccPoint()

' This example returns the Description setting for the
' first Point object in a selection set

On Error Resume Next

' Delete existing SelectionSet
ThisDrawing.SelectionSets("SSet").Delete

' Create the selection set based on a point selection
' and filter for the Point objects
Dim ssetObj As AcadSelectionSet
Set ssetObj = ThisDrawing.SelectionSets.Add("SSet")

Dim gpCode(0) As Integer
Dim dataValue(0) As Variant

gpCode(0) = 0
dataValue(0) = "AECC_POINT"

Dim groupCode As Variant
Dim dataCode As Variant

ThisDrawing.Utility.Prompt vbCrLf & "Select a point object..."

 groupCode = gpCode
dataCode = dataValue
ssetObj.SelectOnScreen groupCode, dataCode

Dim objPoint As AeccPoint
Set objPoint = ssetObj.Item(0)

MsgBox "The value for Description is: " & objPoint.Description, _
  vbCrLfInformation, "Description Example"

End Sub

Sub Example_Description_Alignment()

  ' This example returns the Description for the first Alignment in the collection.
  Dim align As AeccAlignment
  Set align = AeccApplication.ActiveProject.Alignments.Item(0)

  MsgBox "The Description of the first Alignment is: " & align.Description, _
    vbCrLfInformation, "Description Example"

End Sub

Sub Example_Description_Boundary()

  ' This example returns the Description for the first Boundary in the collection.
  Dim surf As AeccSurface
Sub Example_Description_BreakLine()
    ' This example returns the Description
    ' for the first BreakLine in the collection.
    Dim surf As AeccSurface
    Dim brkLine As AeccBreakLine
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
    Set brkLine = surf.Inputs.BreakLines.Item(0)
    MsgBox "The Description for the BreakLine is: " & brkLine.Description, _
        vbInformation, "Description Example"
End Sub

Sub Example_Description_CrossSectionPointCode()
    ' This example returns the point code data for the first cross
    ' section surface in the collection for the first cross section
    ' in the collection.
    Dim aligns As AeccAlignments
    Dim align As AeccAlignment
    Dim xSect As AeccCrossSection
    Dim xSectPCode As AeccCrossSectionPointCode
    Set aligns = AeccApplication.ActiveProject.Alignments
    Set align = aligns.Item(0)
    Set xSect = align.CrossSections.Item(0)
    Set xSectPCode = xSect.CrossSectionPointCodes.Item(0)
End Sub
'Get the alignment name
Dim alignName As String
alignName = align.Name

'Get the cross section station and format it
Dim station As String
station = aligns.DoubleToStaFormat(xSect.station)

MsgBox "The alignment name is: " & alignName & vbCrLf & 
"The first cross section is at station: " & station & vbCrLf & 
"The data for the first point code is: " & vbCrLf & _
vbTab & "Code: " & xSectPCode.Code & vbCrLf & _
vbTab & "Description: " & xSectPCode.Description & vbCrLf & _
vbTab & "Elevation: " & Format(xSectPCode.elevation, "0.00") & vbCrLf & _
vbTab & "Offset: " & Format(xSectPCode.offset, "0.00"), _
vbInformation, "Description Example"

End Sub

Sub Example_Description_PointGroup()

' This function gets the Description for the first PointGroup
' in the collection.
Dim pntGrp As AeccPointGroup
Set pntGrp = AeccApplication.ActiveProject.PointGroups.Item(0)

MsgBox "The Description for the first PointGroup is: " & pntGrp.Description, 
vbInformation, "Description Example"

End Sub

Sub Example_Description_Project()

' This example returns the Description setting for the first Project
' in the collection
Dim proj As AeccProject
Set proj = AeccApplication.Projects.Item(0)
Sub Example_Description_Prototype()

' This example returns the Description for the first prototype in the collection.
Dim prot As AeccPrototype
Set prot = AeccApplication.Prototypes.Item(0)

MsgBox "The Description of the first prototype is: " & prot.Description, _
vbInformation, "Description Example"

End Sub

Sub Example_Description_Surface()

' This example returns the Description for the first surface in the collection.
Dim surf As AeccSurface
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

MsgBox "The Description for the first surface is: " & surf.Description, _
vbInformation, "Description Example"

End Sub
Sub Example_DescriptionFormat()

' This example returns the DescriptionFormat for the first
' DescriptionKey in the first DescriptionKeyFile in the collection.
Dim dKeyFile As AeccDescriptionKeyFile
Dim dKey As AeccDescriptionKey
Set dKeyFile = AeccApplication.ActiveProject.DescriptionKeyFiles.Item(0)
Set dKey = dKeyFile.Item(0)

MsgBox "The value for DescriptionKey DescriptionFormat is " & dKey.DescriptionFormat,
       vbInformation, "DescriptionFormat Example"
End Sub
DescriptionKeyFiles Example

Sub Example_DescriptionKeyFiles()

    ' This example returns the number of DescriptionKeyFiles
    ' in the current project.
    Dim proj As AeccProject
    Set proj = AeccApplication.ActiveProject

    MsgBox "The number of DescriptionKeyFiles in the current Project is: " & proj.DescriptionKeyFiles.Count, vbInformation, "DescriptionKeyFiles Example"

End Sub
DescriptionLayer Example

Sub Example_DescriptionLayer()

    ' This example returns the DescriptionLayer for the first
    ' DescriptionKey in the first DescriptionKeyFile in the collection.
    Dim dKeyFile As AeccDescriptionKeyFile
    Dim dKey As AeccDescriptionKey
    Set dKeyFile = AeccApplication.ActiveProject.DescriptionKeyFiles.Item(0)
    Set dKey = dKeyFile.Item(0)

    MsgBox "The value for DescriptionKey DescriptionLayer is " & dKey.DescriptionLayer,
            vbInformation, "DescriptionLayer Example"

End Sub
Sub Example_DescriptionOverride()

' This function gets the DescriptionOverride for the first PointGroup ' in the collection.
Dim pntGrp As AeccPointGroup
Set pntGrp = AeccApplication.ActiveProject.PointGroups.Item(0)

If pntGrp.DescriptionOverride = True Then
    MsgBox "The DescriptionOverride for the first PointGroup is turned on.", _
    vbInformation, "DescriptionOverride Example"
Else
    MsgBox "The DescriptionOverride for the first PointGroup is turned off.", _
    vbInformation, "DescriptionOverride Example"
End If

End Sub
Sub Example_DescriptionXDRef()

' This function gets the DescriptionXDRef for the first PointGroup
' in the collection.
Dim pntGrp As AeccPointGroup
Set pntGrp = AeccApplication.ActiveProject.PointGroups.Item(0)

MsgBox "The DescriptionXDRef for the first PointGroup is: " & _
    pntGrp.DescriptionXDRef, vbInformation, "DescriptionXDRef Example"

End Sub
Sub Example_DifferenceGrid()

' This example creates a surface from two existing, overlapping, ' surfaces using the Grid method.
Dim surfs As AeccSurfaces
Dim diffSurf As AeccSurface
Set surfs = AeccApplication.ActiveProject.Surfaces

Dim surf0 As String
Dim surf1 As String
Dim surf2 As String
Dim gridx As Double
Dim gridy As Double
Dim pnt As Variant
Dim rows As Integer
Dim columns As Integer

' Get the difference grid input
surf0 = ThisDrawing.Utility.GetString(True, "Enter name for new Surface: ")
surf1 = ThisDrawing.Utility.GetString(True, "Enter name of first Surface: ")
surf2 = ThisDrawing.Utility.GetString(True, "Enter name of overlapping Surface: ")
gridx = ThisDrawing.Utility.GetReal("Enter the width of the grid: ")
gridy = ThisDrawing.Utility.GetReal("Enter the height of the grid: ")
rows = ThisDrawing.Utility.GetInteger("Enter the number of rows in the grid: ")
columns = ThisDrawing.Utility.GetInteger("Enter the number of columns in the ")
pnt = ThisDrawing.Utility.GetPoint(), "Select lower left point of grid: ")

' Create the difference grid
diffSurf = surfs.DifferenceGrid(surf0, surf1, surf2, gridx, gridy, rows, columns)

MsgBox "The number of faces in the new surface is: " & diffSurf.NumberOfFaces
End Sub
## Direction Example

### Examples:

1. **AlignTangent**
2. **ProfileBlock** (Civil Engineering Feature)

---

Sub Example_Direction_AlignTangent()

' This example returns the Direction for the first Tangent found in the first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Tangent entity in the first Alignment."

' Find first Tangent in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kTangent Then
        alignMsg = "The Direction for the first Tangent in the alignment is: " & alignEnt.direction
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "Direction Example"
End Sub

---

Sub Example_Direction_ProfileBlock()

' This example returns the Direction for the first ProfileBlock in the collection
Dim alignProf As AeccProfileBlock
Set alignProf = AeccApplication.ActiveDocument.ProfileBlocks.Item(0)

MsgBox "The Direction for the first ProfileBlock in the collection is: " _ 
& alignProf.Direction, vbInformation, "Direction Example"

End Sub
Sub Example_Documents()

    ' This example returns the number of drawings loaded
    ' in the current session.
    Dim docs As AeccDocuments
    Set docs = AeccApplication.Documents

    MsgBox "The number of loaded drawings is: " & docs.Count, _
        vbInformation, "Documents Example"

End Sub
DoubleToStaFormat Example

Sub Example_DoubleToStaFormat()

' This example returns the DoubleToStaFormat string for 100.0.
Dim aligns As AeccAlignments
Set aligns = AeccApplication.ActiveProject.Alignments

MsgBox "The DoubleToStaFormat for 100.0 is " & aligns.DoubleToStaFormat
    vbInformation, "DoubleToStaFormat Example"

End Sub
Sub Example_DrainsInto()

    ' This example returns the ID of the first WaterShed that is
    ' drained into for the first WaterShed in the collection.
    Dim surf As AeccSurface
    Dim wShed As AeccWaterShed
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
    Set wShed = surf.Outputs.WaterSheds.Item(0)

    MsgBox "The first Watershed Id that is drained into is: " & wShed.Id _
        , vbInformation, "DrainsInto Example"

End Sub
Sub Example_DrawingPath()

    ' This example returns the DrawingPath setting for the first Project
    ' in the collection
    Dim proj As AeccProject
    Set proj = AeccApplication.Projects.Item(0)

    MsgBox "The DrawingPath value for the first Project in the collection is: " & vbCrLf & vbInformation, "DrawingPath Example"

End Sub
Sub Example_Drawings()

' This example returns the number of Drawings
' in the current project.
Dim proj As AeccProject
Set proj = AeccApplication.ActiveProject

MsgBox "The number of Drawings in the current Project is: " & proj.Drawings.Count, vbInformation, "Drawings Example"

End Sub
Sub Example_DrawingSetupPath()

' This example returns the DrawingSetupPath setting.
Dim prefFiles As AeccPreferencesFiles
Set prefFiles = AeccApplication.Preferences.Files

MsgBox "The current value for DrawingSetupPath is: " & prefFiles.DrawingSetupPath, vbInformation, "DrawingSetupPath Example"

End Sub
Easting Example

Examples:

- **AeccPoint**
- **CogoPoint**
- **TinPoint**

---

Sub Example_Easting_AeccPoint()

    ' This example returns the Easting setting for the first Point object in a selection set.
    On Error Resume Next

    ' Delete existing SelectionSet
    ThisDrawing.SelectionSets("SSet").Delete

    ' Create the selection set based on a point selection and filter for the Point objects
    Dim ssetObj As AcadSelectionSet
    Set ssetObj = ThisDrawing.SelectionSets.Add("SSet")

    Dim gpCode(0) As Integer
    Dim dataValue(0) As Variant

    gpCode(0) = 0
    dataValue(0) = "AECC_POINT"

    Dim groupCode As Variant
    Dim dataCode As Variant

    groupCode = gpCode
    dataCode = dataValue
    ssetObj.SelectOnScreen groupCode, dataCode
Dim objPoint As AeccPoint
Set objPoint = ssetObj.Item(0)

MsgBox "The setting for Easting is: " & objPoint.Easting, vbInformation, "Eas
End Sub

Sub Example_Easting_CogoPoint()

' This example returns a Cogo point East coordinate.
Dim dbPoint As AeccCogoPoint
Set dbPoint = AeccApplication.ActiveProject.CogoPoints.Item(0)

MsgBox "The point Easting for the first point in the collection is: " & dbPoint.
   vbInformation, "Easting Example"
End Sub

Sub Example_Easting_TinPoint()

' This example returns the Easting of the first
' TIN Point in the collection.
Dim surf As AeccSurface
Dim tin As AeccTinPoint
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set tin = surf.Outputs.TinPoints.Item(0)

MsgBox "The Easting of the first TIN Point is: " & tin.Easting, vbInformation,
End Sub
Sub Example_EastNorthToXy()

' This example returns the X and Y for a Easting Northing.
Dim util As AeccUtility
Set util = AeccApplication.ActiveDocument.Utility

 Dim var1 As Variant
 Dim ptEN(0 To 2) As Double
 Dim ptXY (0 To 2) As Double
 Dim str As String

  ptEN(0) = 1000
  ptEN(1) = 1000
  ptEN(2) = 0

' Convert point to X, Y
var1 = util.EastNorthToXy(ptEN)

  ptXY(0) = var1(0)
  ptXY(1) = var1(1)
  ptXY(2) = var1(2)

' Make display string
str = Format(ptXY(0), "0.00") + ", " + Format(ptXY(1), "0.00") + ", " + Forma:
MsgBox "Point: " & str , vbInformation, "EastNorthXy Example"

End Sub
Sub Example_Edges()

    ' This example returns the number of Edges in the first
    ' surface in the collection.
    Dim surf As AeccSurface
    Dim surfOut As AeccSurfaceOutputs
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
    Set surfOut = surf.Outputs

    MsgBox "The number of Edges in the first Surface is: " & surfOut.Edges.Count
End Sub
EGPrecision Example

Examples:

1 CrossSectionBlock (Civil Engineering Feature)
1 ProfileBlock (Civil Engineering Feature)

Sub Example_EGPrecision_CrossSectionBlock()

' This example returns the existing ground precision for the
' first alignment cross section in the collection.
Dim alignXsects As AeccCrossSectionBlocks
Dim alignXsect As AeccCrossSectionBlock
Set alignXsects = AeccApplication.ActiveDocument.CrossSectionBlocks
Set alignXsect = alignXsects.Item(0)

' Get the station for the first alignment cross section in the collection
Dim station As String
station = alignXsect.station

MsgBox "The existing ground precision for the alignment cross section at station " & station & " is: " & alignXsect.EGPrecision, vbInformation, "EGPrecision Example"

End Sub

Sub Example_EGPrecision_ProfileBlock()

' This example returns the EGPrecision for the first ProfileBlock
' in the collection
Dim alignProf As AeccProfileBlock
Set alignProf = AeccApplication.ActiveDocument.ProfileBlocks.Item(0)

MsgBox "The EGPrecision for the first ProfileBlock in the collection is: " & & alignProf.EGPrecision, vbInformation, "EGPrecision Example"
End Sub
Sub Example_EGProfiles()

    ' This example returns the number of existing ground profiles for the
    ' first alignment in the collection.
    Dim align As AeccAlignment
    Set align = AeccApplication.ActiveProject.Alignments.Item(0)

    MsgBox "The number of existing ground profiles in first alignment is: " & _
    align.EGProfiles.Count, vbInformation, "EGProfiles Example"

End Sub
Elevation Example

Examples:

- **AeccContour**
- **AeccPoint**
- **CogoPoint**
- **CrossSectionPointCode** (Civil Engineering Feature)
- **ElevationContours**
- **PointGroup**
- **PVI** (Civil Engineering Feature)
- **TinPoint**

```vba
Sub Example_Elevation_AeccContour()

    ' This example displays The Elevation for a selected contour.

    On Error Resume Next
    ' Delete existing SelectionSet
    ThisDrawing.SelectionSets("SSet").Delete

    ' Create the selection set based on a point selection
    ' and filter for Contour objects
    Dim ssetObj As AcadSelectionSet
    Set ssetObj = ThisDrawing.SelectionSets.Add("SSet")

    Dim gpCode(0) As Integer
    Dim dataValue(0) As Variant

    gpCode(0) = 0
```
dataValue(0) = "AECC_CONTOUR"

Dim groupCode As Variant
Dim dataCode As Variant

groupCode = gpCode
dataCode = dataValue
ssetObj.SelectOnScreen groupCode, dataCode

' Get first contour from selection set
Dim objContour As AeccContour
Set objContour = ssetObj.Item(0)

MsgBox "The Contour Elevation is: " & objContour.Elevation, vbInformation,

End Sub

Sub Example_Elevation_AeccPoint()

' This example returns the Elevation setting for the
' first Point object in a selection set

On Error Resume Next

' Delete existing SelectionSet
ThisDrawing.SelectionSets("SSet").Delete

' Create the selection set based on a point selection
' and filter for the Point objects
Dim ssetObj As AcadSelectionSet
Set ssetObj = ThisDrawing.SelectionSets.Add("SSet")

Dim gpCode(0) As Integer
Dim dataValue(0) As Variant

gpCode(0) = 0
dataValue(0) = "AECC_POINT"
Dim groupCode As Variant
Dim dataCode As Variant

groupCode = gpCode
dataCode = dataValue
ssSetObj.SelectOnScreen groupCode, dataCode

Dim objPoint As AeccPoint
Set objPoint = ssSetObj.Item(0)

MsgBox "The setting for Elevation is: " & objPoint.Elevation, vbInformation, 'End Sub

Sub Example_Elevation_CogoPoint()

' This example returns the Elevation for the first CogoPoint in the collection
Dim cogoPnt As AeccCogoPoint
Set cogoPnt = AeccApplication.ActiveProject.CogoPoints.Item(0)

MsgBox "The Elevation for the first CogoPoint in the collection is: " & cogoPnt.Elevation, vbInformation, 'End Sub

Sub Example_Elevation_CrossSectionPointCode()

' This example returns the point code data for the first cross section surface in the collection for the first cross section in the collection.
Dim aligns As AeccAlignments
Dim align As AeccAlignment
Dim xSect As AeccCrossSection
Dim xSectPCode As AeccCrossSectionPointCode
Set aligns = AeccApplication.ActiveProject.Alignments
Set align = aligns.Item(0)
Set xSect = align.CrossSections.Item(0)
Set xSectPCode = xSect.CrossSectionPointCodes.Item(0)

' Get the alignment name
Dim alignName As String
alignName = align.Name

' Get the cross section station and format it
Dim station As String
station = aligns.DoubleToStaFormat(xSect.station)

MsgBox "The alignment name is: " & alignName & vbCrLf & _
"The first cross section is at station: " & station & vbCrLf & _
"The data for the first point code is: " & vbCrLf & _
vbTab & "Code: " & xSectPCode.Code & vbCrLf & _
vbTab & "Description: " & xSectPCode.Description & vbCrLf & _
vbTab & "Elevation: " & Format(xSectPCode.elevation, "0.00") & vbCrLf & _
vbTab & "Offset: " & Format(xSectPCode.offset, "0.00"), _
vbInformation, "Elevation Example"

End Sub

Sub Example_Elevation_ElevationContours()

' This example returns the number of ElevationContours at 100.0 for
' the first surface in the collection
Dim surf As AeccSurface
Dim elevContours As AeccElevationContours
Set surf = AuccApplication.ActiveProject.Surfaces.Item(0)
Set elevContours = Surf.Outputs.ElevationContours

elevContours.Elevation = 100#

MsgBox "The number of ElevationContours at 100.0 is: " & elevContours.Cou
, vbInformation, "Elevation Example"

End Sub
Sub Example_Elevation_PointGroup()

' This function gets the Elevation for the first PointGroup in the collection.
Dim pntGrp As AeccPointGroup
Set pntGrp = AeccApplication.ActiveProject.PointGroups.Item(0)

MsgBox "The Elevation for the first PointGroup is: " & _
pntGrp.Elevation, vbInformation, "Elevation Example"

End Sub

Sub Example_Elevation_PVI()

' This example returns the Elevation value for the second PVI in the first finished ground of the first alignment in the collection.
Dim align As AeccAlignment
Dim FGProf As AeccFGProfile
Dim PVI As AeccPVI
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set FGProf = align.FGProfiles.Item(0)
Set PVI = FGProf.PVIs.Item(1)

MsgBox "The Elevation of the second PVI is: " & _
& Format(PVI.Elevation, "0.000"), vbInformation, "Elevation Example"

End Sub

Sub Example_Elevation_TinPoint()

' This example returns the Elevation of the first TIN Point in the collection.
Dim surf As AeccSurface
Dim tin As AeccTinPoint
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set tin = surf.Outputs.TinPoints.Item(0)
MsgBox "The Elevation of the first TIN Point is: " & tin.Elevation, _
vbInformation, "Elevation Example"

End Sub
ElevationAt Example

Examples:

1. **EGProfile** (Civil Engineering Feature)
2. **FGProfile** (Civil Engineering Feature)

---

Sub Example_ElevationAt_EGProfile()

' This example uses ElevationAt to displays the elevation for a station,
' for the first existing ground profile in the first alignment in the collection.
Dim align As AeccAlignment
Dim EGProf As AeccEGProfile
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set EGProf = align.EGProfiles.Item(0)

Dim station As Double
Dim elevation As Double

' Get the station and the elevation
station = ThisDrawing.Utility.GetReal("Enter a station: ")
elevation = EGProf.ElevationAt(station)

MsgBox "The elevation for station " & Format(station, "0.00") & " is: " _
    & Format(elevation, "0.00"), vbInformation, "ElevationAt Example"
End Sub

---

Sub Example_ElevationAt_FGProfile()

' This example uses ElevationAt to displays the elevation for a station,
' for the finished ground profile in the first alignment in the collection.
Dim align As AeccAlignment
Dim FGProf As AeccFGProfile
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set FGProf = align.FGProfiles.Item(0)
Dim station As Double
Dim elevation As Double

'Get the station and the elevation
station = ThisDrawing.Utility.GetReal("Enter a station: ")
elevation = FGProf.ElevationAt(station)

MsgBox "The elevation for station " & Format(station, "0.00") & " is: " _
     & Format(elevation, "0.000"), vbInformation, "ElevationAt Example"

End Sub
Sub Example_ElevationContours()

' This example returns the number of ElevationContours at 100.0, ' in the first surface in the collection.
Dim surf As AeccSurface
Dim surfOut As AeccSurfaceOutputs
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set surfOut = surf.Outputs

surfOut.ElevationContours.Elevation = 100#

MsgBox "The number of ElevationContours at 100.0 in the first Surface is: " & surfOut.ElevationContours.Count, vbInformation, "ElevationContours Example"
End Sub
Sub Example_ElevationOverride()

' This function gets the Elevation for the first PointGroup
' in the collection.
Dim pntGrp As AeccPointGroup
Set pntGrp = AeccApplication.ActiveProject.PointGroups.Item(0)

If pntGrp.ElevationOverride = True Then
   MsgBox "The ElevationOverride for the first PointGroup is turned on.", vbInformation, "ElevationOverride Example"
Else
   MsgBox "The ElevationOverride for the first PointGroup is turned off.", vbInformation, "ElevationOverride Example"
End If

End Sub
Sub Example_ElevationPrecision()

' This example returns the ElevationPrecision setting for the current drawing.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.Preferences

MsgBox "The current value for ElevationPrecision is: " & dbPref.ElevationPrecision,
        vbInformation, "ElevationPrecision Example"

End Sub
Sub Example_ElevationXDRef()

    ' This function gets the ElevationXDRef for the first PointGroup
    ' in the collection.
    Dim pntGrp As AeccPointGroup
    Set pntGrp = AeccApplication.ActiveProject.PointGroups.Item(0)

    MsgBox "The ElevationXDRef for the first PointGroup is: " & _
        pntGrp.ElevationXDRef, vbInformation, "ElevationXDRef Example"

End Sub
Sub Example_EndDirection()

' This example returns the EndDirection for the first Curve found in the
' first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Curve entity in the first Alignment."

' Find first Curve in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kCurve Then
        alignMsg = "The EndDirection for the first Curve in the alignment is: " &
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "EndDirection Example"

End Sub
EndEasting Example

Examples:

- AlignEntity, AlignCurve, AlignSpiral, AlignTangent
- ParcelEntity, ParcelCurve, ParcelLine

Sub Example_EndEasting_AlignEntity()

    ' This example returns the EndEasting for the first entity in the first Alignment in the collection.
    Dim align As AeccAlignment
    Dim alignEnt As AeccAlignEntity
    Set align = AeccApplication.ActiveProject.Alignments.Item(0)
    Set alignEnt = align.AlignEntities.Item(0)

    MsgBox "The EndEasting of the first entity in the Alignment is: " & alignEnt.EndEasting, vbInformation, "EndEasting Example"

End Sub

Sub Example_EndEasting_ParcelEntity()

    ' This example returns the EndEasting for the first entity in the first Parcel in the collection.
    Dim parcel As AeccParcel
    Dim parcelEnt As AeccParcelEntity
    Set parcel = AeccApplication.ActiveProject.Parcels.Item(0)
    Set parcelEnt = parcel.ParcelEntities.Item(0)

    MsgBox "The EndEasting of the first entity in the Parcel is: " & parcelEnt.EndEasting, vbInformation, "EndEasting Example"

End Sub
EndingStation Example

Examples:

1 AlignEntity, AlignCurve, AlignSpiral, AlignTangent

1 Alignment

1 ProfileBlock (Civil Engineering Feature)

Sub Example_EndingStation_AlignEntity()

' This example returns the EndingStation for the first entity in the first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnt As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set alignEnt = align.AlignEntities.Item(0)

MsgBox "The EndingStation of the first entity in the Alignment is: " & _
alignEnt.EndingStation, vbInformation, "EndingStation Example"

End Sub

Sub Example_EndingStation_Alignment()

' This example returns the EndingStation for the first Alignment in the collection.
Dim align As AeccAlignment
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

MsgBox "The EndingStation of the first Alignment is: " & align.EndingStation, vbInformation, "EndingStation Example"

End Sub

Sub Example_EndingStaion_ProfileBlock()
' This example returns the EndingStation for the first ProfileBlock ' in the collection

Dim alignProf As AeccProfileBlock
Set alignProf = AeccApplication.ActiveDocument.ProfileBlocks.Item(0)

MsgBox "The EndingStation for the first ProfileBlock in the collection is: " _
    & alignProf.EndingStation, vbInformation, "EndingStation Example"

End Sub
EndNorthing Example

Examples:

- AlignEntity, AlignCurve, AlignSpiral, AlignTangent
- ParcelEntity, ParcelCurve, ParcelLine

Sub Example_EndNorthing_AlignEntity()

    ' This example returns the EndNorthing for the first entity in the
    ' first Alignment in the collection.
    Dim align As AeccAlignment
    Dim alignEnt As AeccAlignEntity
    Set align = AeccApplication.ActiveProject.Alignments.Item(0)
    Set alignEnt = align.AlignEntities.Item(0)

    MsgBox "The EndNorthing of the first entity in the Alignment is: " & alignEnt.EndNorthing,
        vbInformation, "EndNorthing Example"

End Sub

Sub Example_EndNorthing_ParcelEntity()

    ' This example returns the EndNorthing for the first entity in the
    ' first Parcel in the collection.
    Dim parcel As AeccParcel
    Dim parcelEnt As AeccParcelEntity
    Set parcel = AeccApplication.ActiveProject.Parcels.Item(0)
    Set parcelEnt = parcel.ParcelEntities.Item(0)

    MsgBox "The EndNorthing of the first entity in the Parcel is: " & parcelEnt.EndNorthing,
        vbInformation, "EndNorthing Example"

End Sub
Sub Example_ExceedBoth()
    ' This example returns the ExceedBoth setting for the WaterSheds.
    Dim surf As AeccSurface
    Dim wSheds As AeccWaterSheds
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
    Set wSheds = surf.Outputs.WaterSheds

    MsgBox "The ExceedBoth setting for WaterSheds is: " & wSheds.ExceedBoth
        vbCrLf, "ExceedBoth Example"
End Sub
Sub Example_ExtEasting()

' This example returns the ExtEasting for the first Spiral found in the
' first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Spiral entity in the first Alignment."

' Find first Spiral in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kSpiral Then
        alignMsg = "The ExtEasting for the first Spiral in the alignment is: " & alignEnt.ExtEasting
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "ExtEasting Example"

End Sub
Sub Example_ExternalSecant()

' This example returns the ExternalSecant for the first Curve found in the
' first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Curve entity in the first Alignment."

' Find first Curve in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kCurve Then
        alignMsg = "The ExternalSecant for the first Curve in the alignment is: " & alignEnt.ExternalSecant
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "ExternalSecant Example"

End Sub
Sub Example_ExternalStaToInternal()

' This example gets the ExternalStaToInternal of 100.0
' for the first Alignment in the collection..
Dim align As AeccAlignment
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim intSta As Variant
intSta = align.ExternalStaToInternal(100#)

MsgBox "The ExternalStaToInternal of 100.0 for the first Alignment is: " & intSta
vbInformation, "ExternalStaToInternal Example"

End Sub
ExtNorthing Example

Sub Example_ExtNorthing()

' This example returns the ExtNorthing for the first Spiral found in the first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Spiral entity in the first Alignment."

' Find first Spiral in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kSpiral Then
        alignMsg = "The ExtNorthing for the first Spiral in the alignment is: " & alignEnt.ExtNorthing
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "ExtNorthing Example"

End Sub
Sub Example_Faces()

    ' This example returns the number of Faces in the first
    ' surface in the collection.
    Dim surf As AeccSurface
    Dim surfOut As AeccSurfaceOutputs
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
    Set surfOut = surf.Outputs

    MsgBox "The number of Faces in the first Surface is: " & surfOut.Faces.Count
        , vbInformation, "Faces Example"

End Sub
FacetDeviation Example

Sub Example_FacetDeviation()

' This example displays the FacetDeviation setting for the current drawing.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.preferences

MsgBox "The setting for FacetDeviation is: " & dbPref.FacetDeviation, _
    vbInformation, "FacetDeviation Example"

End Sub
FGPrecision Example

Examples:

1 [CrossSectionBlock](Civil Engineering Feature)
1 [ProfileBlock](Civil Engineering Feature)

Sub Example_FGPrecision_CrossSectionBlock()

' This example returns the finished ground precision for the
' first alignment cross section in the collection.
Dim alignXSEcts As AeccCrossSectionBlocks
Dim alignXSEct As AeccCrossSectionBlock
Set alignXSEcts = AeccApplication.ActiveDocument.CrossSectionBlocks
Set alignXSEct = alignXSEcts.Item(0)

'Get the station for the first alignment cross section in the collection
Dim station As String
station = alignXSEct.station

MsgBox "The finished ground precision for the alignment cross section at station " & station & " is: 
alignXSEct.FGPrecision, vbInformation, "FGPrecision Example"
End Sub

Sub Example_FGPrecision_ProfileBlock()

' This example returns the FGPrecision for the first ProfileBlock
' in the collection
Dim alignProf As AeccProfileBlock
Set alignProf = AeccApplication.ActiveDocument.ProfileBlocks.Item(0)

MsgBox "The FGPrecision for the first ProfileBlock in the collection is: ", _
& alignProf.FGPrecision, vbInformation, "FGPrecision Example"
End Sub
Sub Example_FGProfiles()

' This example returns the number of finished ground profiles for the
' first alignment in the collection.
Dim align As AeccAlignment
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

MsgBox "The number of finished ground profiles in first alignment is: "
& align.FGProfiles.Count , vbInformation, "FGProfiles Example"

End Sub
Sub Example_File()

    ' This example returns the File for the first FileLock in the collection.
    Dim filelock As AeccFileLock
    Set filelock = AeccApplication.ActiveProject.fileLocks.Item(0)

    MsgBox "The File of the first FileLock in the collection is: " & filelock.File, _
    vbInformation, "File Example"

End Sub
FileLocks Example

Sub Example_FileLocks()

    ' This example returns the number of FileLocks
    ' in the current project.
    Dim proj As AeccProject
    Set proj = AeccApplication.ActiveProject

    MsgBox "The number of FileLocks in the current Project is: " & proc.FileLocks.Count, vbInformation, "FileLocks Example"

End Sub
**Files Example**

*Sub Example_Files()*

' This example returns the current setting of PrototypePath ' from the preferences object.

```vbnet
Dim prefs As AeccPreferences
Set prefs = AeccApplication.Preferences

MsgBox "The current value for PrototypePath is: " & prefs.Files.PrototypePath:
    vbInformation, "Files Example"

End Sub
```
Sub Example_FindAllConnectingEdges()

' This example returns all the connecting edges for a given point in the first surface in the surface collection
Dim surf As AeccSurface
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

' Get point for calculating connecting edges
Dim pnt As Variant
pnt = ThisDrawing.Utility.GetPoint(, "Select point in the surface: ")

' Get an actual surface point.
pnt = surf.FindPoint(pnt(0), pnt(1))

' Get edge points
Dim edgePnts As Variant
edgePnts = surf.FindAllConnectingEdges(pnt(0), pnt(1))

Dim lineObj As AcadLine
Dim pnt1(0 To 2) As Double
Dim pnt2(0 To 2) As Double
Dim pnt3(0 To 2) As Double

' Draw red lines around first triangle definition
pnt1(0) = edgePnts(0)
pnt1(1) = edgePnts(1)
pnt1(2) = edgePnts(2)
pnt2(0) = edgePnts(3)
pnt2(1) = edgePnts(4)
pnt2(2) = edgePnts(5)
pnt3(0) = edgePnts(6)
pnt3(1) = edgePnts(7)
pnt3(2) = edgePnts(8)
Set lineObj = ThisDrawing.ModelSpace.AddLine(pnt1, pnt2)
lineObj.Color = acRed
Set lineObj = ThisDrawing.ModelSpace.AddLine(pnt2, pnt3)
lineObj.Color = acRed
Set lineObj = ThisDrawing.ModelSpace.AddLine(pnt3, pnt1)
lineObj.Color = acRed

End Sub
Sub Example_FindAllFaces()

' This example begins by getting a faces collection for an entered
' point in the first surface in the collection. Then the FindAllFaces
' is used to get the entire faces collection.
Dim surf As AeccSurface
Dim faces As AeccFaces
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set faces = surf.Outputs.faces

' Return a point using a prompt
Dim returnPnt As Variant
returnPnt = ThisDrawing.Utility.GetPoint(, "Select a point: ")

' Find face points
faces.FindFace returnPnt(0), returnPnt(1)

MsgBox "The number of Faces in the collection after FindFace is: " & faces.Count,
    vbInformation, "FindAllFaces Example"

faces.FindAllFaces

MsgBox "The number of Faces in the collection after FindAllFaces is: " & faces.Count,
    vbInformation, "FindAllFaces Example"

End Sub
Sub Example_FindConnectingEdge()

' This example returns the connecting edge for a given point
' in the first surface in the collection.
Dim surf As AeccSurface
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

' Get point for calculating connecting edges
Dim pnt As Variant
pnt = ThisDrawing.Utility.GetPoint(, "Select point in the surface: ")

' Get edge points
Dim edgePnts As Variant
edgePnts = surf.FindConnectingEdge(pnt(0), pnt(1))

Dim lineObj As AcadLine
Dim startPnt(0 To 2) As Double
Dim endPnt(0 To 2) As Double

' Draw a red line along edge
startPnt(0) = edgePnts(0)
startPnt(1) = edgePnts(1)
startPnt(2) = edgePnts(2)
endPnt(0) = edgePnts(3)
endPnt(1) = edgePnts(4)
endPnt(2) = edgePnts(5)
Set lineObj = ThisDrawing.ModelSpace.AddLine(startPnt, endPnt)
lineObj.Color = acRed

End Sub
Sub Example_FindFace()

' This example returns a new face collection for an entered point in
' the first surface in the collection.
Dim surf As AeccSurface
Dim faces As AeccFaces
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set faces = Surf.Outputs.Faces

MsgBox "The number of Faces in the collection before FindFace is: " & faces.Count, vbInformation, "FindFace Example"

' Return a point using a prompt
Dim returnPnt As Variant
returnPnt = ThisDrawing.Utility.GetPoint(, "Select a point: ")

' Find face points
faces.FindFace returnPnt(0), returnPnt(1)

MsgBox "The number of Faces in the collection after FindFace is: " & faces.Count, vbInformation, "FindFace Example"

End Sub
Sub Example_FindPath()

' This example returns the face collection for an entered path in
' the first surface in the collection.
Dim surf As AeccSurface
Dim faces As AeccFaces
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set faces = surf.Outputs.faces

MsgBox "The number of Faces in the collection before FindPath is: " & faces.Count, vbInformation, "FindPath Example"

' Select the path for faces
Dim pnt1 As Variant
Dim pnt2 As Variant
pnt1 = ThisDrawing.Utility.GetPoint(, "Select first point on a surface: ")
pnt2 = ThisDrawing.Utility.GetPoint(pnt1, "Select second point on a surface: ")

faces.FindPath(pnt1(0), pnt1(1), pnt2(0), pnt2(1))

MsgBox "The number of Faces in the collection before FindPath is: " & faces.Count, vbInformation, "FindPath Example"

End Sub
Sub Example_FindPoint()

' This example draws a point at the surface point for a selected point
' in the first surface in the collection. The point id drawn with the default
' point style.
Dim surf As AeccSurface
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

' Get point for calculating point
Dim pnt As Variant
pnt = ThisDrawing.Utility.GetPoint(, "Select point in the surface: ")

' Get surface point
Dim edgePnts As Variant
tDim edgePnts = surf.FindPoint(pnt(0), pnt(1))

Dim pntObj As AcadPoint
Dim startPnt(0 To 2) As Double

' Draw a red point at found point
startPnt(0) = edgePnts(0)
startPnt(1) = edgePnts(1)
startPnt(2) = edgePnts(2)
Set pntObj = ThisDrawing.ModelSpace.AddPoint(startPnt)
pntObj.Color = acRed

End Sub
FirstTimeDrawingSetup Example

Sub Example_FirstTimeDrawingSetup()

' This example returns the FirstTimeDrawingSetup setting.
Dim prefUser As AeccPreferencesUser
Set prefUser = AeccApplication.Preferences.User

' Convert the constant to a string.
Dim strType As String
If prefUser.FirstTimeDrawingSetup = kDrawingSetupWizard Then
    strType = "Wizard"
End If
If prefUser.FirstTimeDrawingSetup = kDrawingSetupCommand Then
    strType = "Command"
End If
If prefUser.FirstTimeDrawingSetup = kAutoloadSetupFile Then
    strType = "Autoload"
End If

MsgBox "The current value for FirstTimeDrawingSetup is: " & strType, _
    vbInformation, "FirstTimeDrawingSetup Example"

End Sub
Sub Example_FirstTimeDrawingSetupFile()

' This example returns the FirstTimeDrawingSetupFile setting.
Dim prefUser As AeccPreferencesUser
Set prefUser = AeccApplication.PREFERENCES.User

MsgBox "The current value for FirstTimeDrawingSetupFile is: " & _
prefUser.FirstTimeDrawingSetupFile, vbInformation, "FirstTimeDrawingSetupFile Example"

End Sub
Sub Example_Format()

    ' This example returns the Format setting for the first PointFile
    ' in the collection.
    Dim surf As AeccSurface
    Dim pntFile As AeccPointFile
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
    Set pntFile = surf.Inputs.PointFiles.Item(0)

    MsgBox "The Format for the first PointFile in the collection is: " & pntFile.Format, vbInformation, "Format Example"

End Sub
Sub Example_FormatsPath()

' This example returns the FormatsPath setting.
Dim prefFiles As AeccPreferencesFiles
Set prefFiles = AeccApplication.Preferences.Files

MsgBox "The current value for FormatsPath is: " & prefFiles.FormatsPath, _
    vbInformation, "FormatsPath Example"

End Sub
Sub Example_FullDescription()

' This example returns the FullDescription for the first CogoPoint in the collect
Dim cogoPnt As AeccCogoPoint
Set cogoPnt = AeccApplication.ActiveProject.CogoPoints.Item(0)

MsgBox "The FullDescription for the first CogoPoint in the collection is: " & _
cogoPnt.FullDescription, vbInformation, "FullDescription Example"

End Sub
**FullName Example**

**Examples:**

1. **DescriptionKeyFile**
2. **Drawing**
3. **Project**

---

_{Sub} Example_FullName_DescriptionKeyFile()_{

' This example returns the FullName for the first DescriptionKeyFile in the collection.
Dim dKeyFile As AeccDescriptionKeyFile
Set dKeyFile = AeccApplication.ActiveProject.DescriptionKeyFiles.Item(0)

MsgBox "The value for DescriptionKeyFile FullName is " & dKeyFile.FullName, vbInformation, "FullName Example"

End Sub

---

{Sub} Example_FullName_Drawing()_{

' This example returns the FullName setting for the first Drawing in the collection
Dim dwg As AeccDrawing
Set dwg = AeccApplication.ActiveProject.Drawings(0)

MsgBox "The FullName for the first Drawing in the collection is: " & dwg.FullName, vbInformation, "FullName Example"

End Sub

---

{Sub} Example_FullName_Project()_{

---
'This example returns the FullName setting for the first Project in the collection

Dim proj As AeccProject
Set proj = AeccApplication.Projects.Item(0)

MsgBox "The FullName value for the first Project in the collection is: " & proj.FullName,
vbInformation, "FullName Example"

End Sub
Sub Example_GetBoundingBox()

' This example returns the two points defining the Bounding Box
' for the first surface in the collection.
Dim surf As AeccSurface
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

Dim pntLL As Variant
Dim pntUR As Variant

' Build the surface and get the bounding box points
surf.Build
surf.GetBoundingBox pntLL, pntUR

MsgBox "Lower left bounding box point: " & pntLL(0) & ", " & pntLL(1) & ", " & pntLL(2) & vbCrLf & vbCrLf & "Upper right bounding box point: " & pntUR(0) & ", " & pntUR(1) & ", " & pntUR(2) & vbCrLf, vbInformation, "GetBoundingBox Example"

End Sub
Sub Example_GetDouble()

' This example uses GetDouble to return the ContLabelSpacingDist ' for PreferencesSurface.
DimsurfPref As AeccPreferencesSurface
Set surfPref = AeccApplication.ActiveProject.preferences.Surface

MsgBox "The value for ContLabelSpacingDist is " & surfPref.GetDouble(kContLabelSpacingDist), vbInformation, "GetDouble Example"

End Sub
Sub Example_GetElevation()

' This example returns the Elevation for an entered point for the first
' surface in the collection.
Dim surfs As AeccSurfaces
Dim surf As AeccSurface
Set surfs = AeccApplication.ActiveProject.Surfaces
Set surf = surfs.Item(0)
Dim returnPnt As Variant
Dim elev As Double

' Get a point
returnPnt = ThisDrawing.Utility.GetPoint(, "Select a point: ")

' Get the elevation from the selected point
elev = surf.GetElevation(returnPnt(0), returnPnt(1))

MsgBox "The elevation from GetElevation is " & elev, vbInformation, "GetElev"
Sub Example_GetInteger()

    ' This example uses GetInteger to return the ContLabelPrecision
    ' for PreferencesSurface.
    Dim surfPref As AeccPreferencesSurface
    Set surfPref = AeccApplication.ActiveProject.preferences.Surface

    MsgBox "The value for ContLabelPrecision is " & surfPref.GetInteger(kContL
        vbInformation, "GetInteger Example"

End Sub
GetLayerName Example

Examples:

1 CrossSectionBlock (Civil Engineering Feature)

1 ProfileBlock (Civil Engineering Feature)

Sub Example_GetLayerName_CrossSectionBlock()

    ' This example uses GetLayerName to get the grid layer name for the
    ' first cross section in the first alignment profile in the collection.
    Dim alignXSec As AeccCrossSectionBlock
    Set alignXSec = AeccApplication.ActiveDocument.CrossSectionBlocks.Item(0)

    MsgBox "The existing ground layer for the first alignment cross section in the
    & alignXSec.GetLayerName(kCrossSectionEGLayer), vbInformation, "Get

End Sub

Sub Example_GetLayerName_ProfileBlock()

    ' This example uses GetLayerName to get the grid layer name for the
    ' first alignment profile in the collection.
    Dim alignProf As AeccAlignProfile
    Set alignProf = AeccApplication.ActiveDocument.ProfileBlocks.Item(0)

    MsgBox "The grid layer for the first ProfileBlock in the collection is: " _
    & alignProf.GetLayerName(kGrid), vbInformation, "GetLayerName Exam"
Sub Example_GetStaStrWithEquation()

' This example gets the GetStaStrWithEquations from a user supplied station for the first Alignment in the collection.
Dim Align As AeccAlignment
Set Align = AeccApplication.ActiveProject.Alignments.Item(0)
Dim staEqu As String
Dim Station As Double

' Get the station
Station = ThisDrawing.Utility.GetReal("Enter the station: ")

' Get the station with Station Equations applied
staEqu = Align.GetStaStrWithEquations(Station)

MsgBox "The GetStaStrWithEquation value for station " & Station & " is: " & staEqu,
End Sub
Sub Example_GetStaWithEquation()

' This example gets the GetStaWithEquations from a user supplied station
' for the first Alignment in the collection.
Dim align As AeccAlignment
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Dim staEqu As Double
Dim station As Double

' Get the station
station = ThisDrawing.Utility.GetReal("Enter the station: ")

' Get the station with Station Equations applied
staEqu = align.GetStaWithEquations(station)

MsgBox "The GetStaWithEquation value for station " & station & " is: " & staEqu

End Sub
Sub Example_GetString()

    ' This example uses GetString to return the SurfaceLayer
    ' for PreferencesSurface.
    Dim surfPref As AeccPreferencesSurface
    Set surfPref = AeccApplication.ActiveProject.preferences.Surface

    MsgBox "The value for SurfaceLayer is " & surfPref.GetString(kSurfaceLayer)

End Sub
Sub Example_GridEasting()

' This example returns the GridEasting for the first CogoPoint in the collection.
Dim cogoPnt As AeccCogoPoint
Set cogoPnt = AeccApplication.ActiveProject.CogoPoints.Item(0)

MsgBox "The GridEasting for the first CogoPoint in the collection is: " & cogoPnt.GridEasting, vbInformation, "GridEasting Example"

End Sub
Sub Example_GridNorthing()

' This example returns the GridNorthing for the first CogoPoint in the collection
Dim cogoPnt As AeccCogoPoint
Set cogoPnt = AeccApplication.ActiveProject.CogoPoints.Item(0)

MsgBox "The GridNorthing for the first CogoPoint in the collection is: " & cogoPnt.GridNorthing, vbInformation, "GridNorthing Example"

End Sub
Examples:

1 CogoPoint

1 PointGroup

Sub Example_GroupName_CogoPoint()

' This example returns the GroupName for the first CogoPoint in the collection
' This example assumes the first point in the collection is in a
' Point Group named "Example Group".
Dim cogoPnts As AeccCogoPoints
Dim cogoPnt As AeccCogoPoint
Set cogoPnts = AeccApplication.ActiveProject.CogoPoints

' Filter CogoPoints collection on GroupName
cogoPnts.GroupName = "Example Group"

Set cogoPnt = cogoPnts.Item(0)

MsgBox "The GroupName for the first CogoPoint in the collection is: " & cogoPnt.Groupname, vbInformation, "GroupName Example"
End Sub

Sub Example_GroupName_PointGroup()

' This function gets the GroupName for the first PointGroup
' in the collection.
Dim pntGrp As AeccPointGroup
Set pntGrp = AeccApplication.ActiveProject.PointGroups.Item(0)

MsgBox "The GroupName for the first PointGroup is: " & _
pntGrp.GroupName, vbInformation, "GroupName Example"
End Sub
Sub Example_Height()

' This example returns the height for the
' first alignment cross section in the collection.
Dim alignXsects As AeccCrossSectionBlocks
Dim alignXsect As AeccCrossSectionBlock
Set alignXsects = AeccApplication.ActiveDocument.CrossSectionBlocks
Set alignXsect = alignXsects.Item(0)

'Get the station for the first alignment cross section in the collection
Dim station As String
station = alignXsect.station

MsgBox "The height for the alignment cross section at station " & station & " is:
          Format(alignXsect.Height, "0.00"), vbInformation, "Height Example"
End Sub
Sub Example_HelpPath()

' This example returns the HelpPath setting.
Dim prefFiles As AeccPreferencesFiles
Set prefFiles = AeccApplication.Preferences.Files

MsgBox "The current value for HelpPath is: " & prefFiles.HelpPath, _
   vbInformation, "HelpPath Example"

End Sub
ID Example

Examples:

- Boundary
- Breakline
- ContourItem
- WaterShed

---

Sub Example_Id_Boundary()

    ' This example returns the Id
    ' for the first Boundary in the collection.
    Dim surf As AeccSurface
    Dim bound As AeccBoundary
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
    Set bound = surf.Inputs.Boundaries.Item(0)

    MsgBox "The Id for the Boundary is: " & bound.Id &, vbInformation, "Id Example"

End Sub

---

Sub Example_Id_BreakLine()

    ' This example returns the Id
    ' for the first BreakLine in the collection.
    Dim surf As AeccSurface
    Dim brkLine As AeccBreakLine
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
    Set brkLine = surf.Inputs.BreakLines.Item(0)

    MsgBox "The Id for the BreakLine is: " & brkLine.Id &, vbInformation, "Id Example"
Sub Example_Id_ContourItem()

' This example returns the Id
' for the first ContourItem in the collection.
Dim surf As AeccSurface
Dim cont As AeccContourItem
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set cont = Surf.Inputs.ContourItems.Item(0)

MsgBox "The Id for the ContourItem is: " & cont.Id _, vbInformation, "Id Example"
End Sub

Sub Example_Id_Watershed()

' This example returns the ID of the first WaterShed in the collection.
Dim surf As AeccSurface
Dim wShed As AeccWaterShed
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set wShed = surf.Outputs.WaterSheds.Item(0)

MsgBox "The Id for the first WaterShed in the collection is: " & wShed.Id _, vbInformation, "Id Example"
End Sub
Import Example

Examples:

1 **Alignment**

1 **CrossSection** (Civil Engineering Feature)

1 **EGProfile** (Civil Engineering Feature)

1 **Parcel**

1 **Surface**

---

`Sub Example_Import_Alignment()`

' This example adds an Alignment made up of a tangent.
Dim aligns As AeccAlignments
Dim align As AeccAlignment
Set aligns = AeccApplication.ActiveProject.Alignments

' Add an Alignment named "Example Alignment" and starting at Station 50.0
Set align = aligns.Add("Example Alignment", 50#)

' Add a tangent
Dim tangent As AeccAlignTangent
Set tangent = align.AddTangent(0#, 0#, 150#, 0#)

' Import the alignment to the drawing
align.Import

MsgBox "The total number of entities in the Alignment is: " & align.AlignEntities.Count,
vbInformation, "Import Example"

`End Sub`

---

`Sub Example_Import_CrossSection()`
' This example inserts the first cross section in the
cross section collection for the first alignment in the collection.
Dim aligns As AeccAlignments
Dim align As AeccAlignment
Dim xSect As AeccCrossSection
Set aligns = AeccApplication.ActiveProject.Alignments
Set align = aligns.Item(0)
Set xSect = align.CrossSections.Item(0)

' Get the cross section station and format it
Dim station As String
station = aligns.DoubleToStaFormat(xSect.station)

' Get the insertion point for the cross section
Dim returnPnt As Variant
returnPnt = ThisDrawing.Utility.GetPoint_
    (, "Select bottom insertion point for the cross section at station " & station & ')

' Import cross section with complete geometry
xSect.import returnPnt, False

End Sub

Sub Example_Import_EGProfile()

' This example imports the the first existing ground profile
' in the first alignment in the collection.
Dim aligns As AeccAlignments
Dim align As AeccAlignment
Dim EGProf As AeccEGProfile
Set aligns = AeccApplication.ActiveProject.Alignments
Set align = aligns.Item(0)
Set EGProf = align.EGProfiles.Item(0)

'Set the name of the first alignment to the current alignment
aligns.CurrentAlignment = align.Name
'Determine default starting and ending station
Dim min As Double
Dim asta As Variant
asta = EGProf.StationElevations
min = asta(1)

Dim i As Integer
For i = 0 To UBound(asta) - 1 Step 2
    If asta(i + 1) < min Then
        min = asta(i + 1)
    End If
Next

'Get the start point for the existing ground profile
Dim returnPnt As Variant
returnPnt = ThisDrawing.Utility.GetPoint(, "Select a starting point:")

On Error Resume Next

'Get gating station
Dim startSta As Double
startSta = asta(0)
startSta = ThisDrawing.Utility.GetReal("Starting Station <" & startSta & ">: ")

'Get ending station
Dim endSta As Double
endSta = asta(UBound(asta) - 1)
endSta = ThisDrawing.Utility.GetReal("Ending Station: <" & endSta & ">: ")

'Get datum elevation
Dim datumElev As Double
datumElev = min
datumElev = ThisDrawing.Utility.GetReal("Datum Elevation: <" & datumElev

'Get vertical scale
Dim vertScale As Double
Dim pref As AeccDatabasePreferences
Set pref = AeccApplication.ActiveDocument.Preferences
vertScale = pref.VerticalScale
vertScale = ThisDrawing.Utility.GetReal("Vertical Scale: <" & vertScale & ">:"

'Get direction
Dim dir As Boolean
dir = True
Dim ans As String
ans = ThisDrawing.Utility.GetString(0, "Direction LeftToRight (Y/N): ")
dir = Switch(ans = "Y", True, ans = "y", True, ans = "N", False, ans = "n", False

'Get block only
Dim block As Boolean
block = True
Dim ans As String
ans = ThisDrawing.Utility.GetString(0, "Block only (Y/N): ")
block = Switch(ans = "Y", True, ans = "y", True, ans = "N", False, ans = "n", False

'Import the existing ground profile
EGProf.import returnPnt, startSta, endSta, datumElev, vertScale, dir, block

End Sub

Sub Example.Import.Parcel()

' This example adds a Parcel made up of lines and an arc.
' The parcel is then imported into the drawing.
Dim parcels As AeccParcels
Dim parcel As AeccParcel
Set parcels = AeccApplication.ActiveProject.parcels

' Add a new Parcel
Set parcel = parcels.Add("New Parcel")

' Add lines and a curve to the Parcel
parcel.AddLine 50#, 50#, 150#, 50#
parcel.AddLine 150#, 50#, 150#, 200#
parcel.AddCurve 150#, 200#, 100#, 200#, 50#, 200#, True
parcel.AddLine 50#, 200#, 50#, 50#

' Import the new Parcel
parcel.Import

MsgBox "The total number of entities in the parcel is: " & parcel.ParcelEntities.Count,
vbInformation, "Import Example"

End Sub

Sub Example_Import_Surface()

' This example draws the first surface in the collection.
Dim surf As AeccSurface
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

surf.Import

MsgBox "The Name of the first surface just imported is: " & surf.Name,
vbInformation, "Import Example"

End Sub
Sub Example_Inputs()

' This example returns the Count of Pointfiles in the first Surface
' in the collection by using the Inputs property.
Dim surf As AeccSurface
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

MsgBox "The number of PointFiles in the first surface is: " & surf.Inputs.PointFiles.Count, vbInformation, "Inputs Example"

End Sub
InstantGrade Example

Examples:

- **EGProfile** (Civil Engineering Feature)
- **FGProfile** (Civil Engineering Feature)

---

Sub Example_InstantGrade_FGProfile()

' This example gets a station from the user and returns the grade for
' the first existing ground profile in the first alignment in the collection.
Dim align As AeccAlignment
Dim EGProf As AeccEGProfile
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set EGProf = align.EGProfiles.Item(0)

Dim station As Double
Dim grade As Double
Dim prompt As String

' Format the prompt and get the station
prompt = "Enter a station on alignment " & align.Name & ": 
station = ThisDrawing.Utility.GetReal(prompt)

' Get the grade
EGProf.InstantGrade station, grade

MsgBox "The grade at station and format the prompt" & Format(station, "0.00")
   & Format(grade, "0.00") & ",", vbInformation, "InstantGrade Example"

End Sub

---

Sub Example_InstantGrade_FGProfile()

' This example gets a station from the user and returns the grade and difference
' for the first finished ground profile in the first alignment in the collection.
Dim align As AeccAlignment
Dim FGProf As AeccFGProfile
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set FGProf = align.FGProfiles.Item(0)

Dim station As Double
Dim grade As Double
Dim diff As Double
Dim prompt As String

'Format the prompt and get the station
prompt = "Enter a station on alignment " & align.Name & ": 
station = ThisDrawing.Utility.GetReal(prompt)

'Get the grade and difference
FGProf.InstantGrade station, grade, diff

MsgBox "The grade at station " & Format(station, "0.00") & ", " & 
% and the difference is " & Format(diff, "0.00") & "%.

vbInformation, "InstantGrade Example"

End Sub
Sub Example_IsBreakline()

    ' This example returns the IsBreakLine setting
    ' for the first Boundary in the collection.
    Dim surf As AeccSurface
    Dim bound As AeccBoundary
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
    Set bound = surf.Inputs.Boundaries.Item(0)

    MsgBox "The IsBreakline setting for the Boundary is: " & bound.IsBreakLine
        , vbInformation, "IsBreakLine Example"

End Sub
Sub Example_IsNameSupported()

' This gets the IsNameSupported for the CogoPoints collection
Dim cogoPnts As AeccCogoPoints
Set cogoPnts = AeccApplication.ActiveProject.CogoPoints

MsgBox "The IsNameSupported for CogoPoints is: " & cogoPnts.IsNameSupported, vbInformation, "IsNameSupported Example"

End Sub
Sub Example_IsVisible()

    ' This example returns the visibility setting for the 
    ' first face item in the collection.
    Dim surf As AeccSurface
    Dim face As AeccFace
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
    Set face = surf.Outputs.Faces.Item(0)

    ' Checks if the Face is visible
    If face.IsVisible = True Then
        MsgBox "The first Face in the collection is Visible!", vbInformation, "IsVisible Example"
    Else
        MsgBox "The first Face in the collection is Invisible!", vbInformation, "IsVisible Example"
    End If

End Sub
Sub Example_Item()

' This example show two uses of the Item method.
' The first uses Item with an index counter to return an item in a collection.
' The second uses Item with a string to return an item in a collection.

' Iterate through the model space collection,
' get all the items in the collection
' and store them in an array called newObjs.
Dim count As Integer
count = AeccApplication.Projects.count

ReDim newObjs(count) As AeccProjects
Dim index As Integer
For index = 0 To count – 1
    Set newObjs(index) = AeccApplication.Projects.Item(index)
Next

' Get a particular item, in this case a project, based on name "shaker".
Dim proj As AeccProject
Set proj = AeccApplication.Projects.Item("shaker")

End Sub
Sub Example_K()

' This example returns the K value for the first Spiral found in the first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Spiral entity in the first Alignment."

' Find first Spiral in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kSpiral Then
        alignMsg = "The K value for the first Spiral in the alignment is: " & align.
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "K Example"

End Sub
Keywords Example

Sub Example_Keywords()

    ' This example returns the Keywords setting for the first Project
    ' in the collection
    Dim proj As AeccProject
    Set proj = AeccApplication.Projects.Item(0)

    MsgBox "The Keywords value for the first Project in the collection is: " & proj.vbInformation, "Keywords Example"

End Sub
Sub Example_Label()

    ' This example returns the Label for the first FileLock in the collection.
    Dim filelock As AeccFileLock
    Set filelock = AeccApplication.ActiveProject.fileLocks.Item(0)

    MsgBox "The Label of the first FileLock in the collection is: " & filelock.Label,
            vbInformation, "Label Example"

End Sub
Sub Example_LabelPoints()

' This example displays a LabelPoint for a selected contour.
On Error Resume Next

' Delete existing SelectionSet
ThisDrawing.SelectionSets("SSet").Delete

' Create the selection set based on a point selection
' and filter for Contour objects
Dim ssetObj As AcadSelectionSet
Set ssetObj = ThisDrawing.SelectionSets.Add("SSet")

Dim mode As Integer
Dim gpCode(0) As Integer
Dim dataValue(0) As Variant

gpCode(0) = 0
dataValue(0) = "AECC_CONTOUR"

Dim groupCode As Variant
Dim dataCode As Variant
Dim returnPnt As Variant

groupCode = gpCode
dataCode = dataValue
returnPnt = ThisDrawing.Utility.GetPoint(, "Select a contour line: ")
ssetObj.SelectAtPoint returnPnt, groupCode, dataCode

Dim objContour As AeccContour
Set objContour = ssetObj.Item(0)
Dim coords As Variant

coords = objContour.LabelPoints
MsgBox "The first point in LabelPoints is " & coords(0) & ", " & coords(0) & vbCrLf & "vbInformation, " & quotation & "LabelPoints Example""
Sub Example_LabelStyle()

' This function gets the label style override for the project database point group
Dim ptGrp As AeccPointGroup
Set ptGrp = AeccApplication.ActiveProject.PointGroups.Item(0)

' Initialize message to no label style override defined
Dim labelMsg As String
labelMsg = "There is no label style override for point group " & _
            ptGrp.GroupName & "."

' Test if label style is defined and if true define new message
If ptGrp.LabelStyle "" Then
    labelMsg = "The label style override for point group " & _
               ptGrp.GroupName & " is: " & ptGrp.LabelStyle
End If

MsgBox labelMsg, vbInformation, "LabelStyle Example"

End Sub
LabelStyleOverride Example

Sub Example_LabelStyleOverride()

' This function determines if the point group label style is overridden.
Dim ptGrp As AeccPointGroup
Set ptGrp = AeccApplication.ActiveProject.PointGroups.Item(0)

MsgBox "Label style override for the project database point group is set to: " & ptGrp.LabelStyleOverride, vbInformation, "LabelStyleOverride Example"

End Sub
LabelStylePath Example

Sub Example_LabelStylePath()

    ' This example returns the LabelStylePath setting.
    Dim prefFiles As AeccPreferencesFiles
    Set prefFiles = AeccApplication.Preferences.Files

    MsgBox "The current value for LabelStylePath is: " & prefFiles.LabelStylePath,
            vbInformation, "LabelStylePath Example"

End Sub
Sub Example_LabelStyleXDRef()

' This function gets the label style XDRef override for the
' project database point group
Dim ptGrp As AeccPointGroup
Set ptGrp = AeccApplication.ActiveProject.PointGroups.Item(0)

' Initialize message to no label style XDRef override defined
Dim labelMsg As String
labelMsg = "There is no label style XDRef override for point group " & _
  ptGrp.GroupName & "."

' Test if XDRef label style is defined and if true define new message
If ptGrp.LabelStyleXDRef = "" Then
  labelMsg = "The label style XDRef override for point group " & _
    ptGrp.GroupName & " is: " & ptGrp.LabelStyleXDRef
End If

MsgBox labelMsg, vbInformation, "LabelStyleXDRef Example"

End Sub
Sub Example_LastUsedDwg()

    ' This example returns the LastUsedDwg setting.
    Dim prefUser As AeccPreferencesUser
    Set prefUser = AeccApplication.Preferences.User

    MsgBox "The current value for LastUsedDrawing is: " & prefUser.LastUsedDwg,
        vbInformation, "LastUsedDwg Example"

End Sub
Sub Example_LastUsedDwgPath()

' This example returns the LastUsedDwgPath setting.
Dim prefUser As AeccPreferencesUser
Set prefUser = AeccApplication.Preferences.User

MsgBox "The current value for LastUsedDwgPath is: " & prefUser.LastUsedDwgPath
     vbCrLfInformation, "LastUsedDwgPath Example"

End Sub
Sub Example_LastUsedProj()

    ' This example returns the LastUsedProj setting.
    Dim prefUser As AeccPreferencesUser
    Set prefUser = AeccApplication.Preferences.User

    MsgBox "The current value for LastUsedProj is: " & prefUser.LastUsedProj, _
        vbInformation, "LastUsedProj Example"

End Sub
Sub Example_LastUsedProjPath()

' This example returns the LastUsedProjPath setting.
Dim prefUser As AeccPreferencesUser
Set prefUser = AeccApplication.Preferences.User

MsgBox "The current value for LastUsedProjPath is: " & prefUser.LastUsedProjPath, vbInformation, "LastUsedProjPath Example"

End Sub
Sub Example_Latitude ()

' This example returns the Latitude and Longitude
' for the first CogoPoint in the collection.
Dim cogoPnt As AeccCogoPoint
Set cogoPnt = AeccApplication.ActiveProject.CogoPoints.Item(0)

MsgBox "The Latitude for the first CogoPoint in the collection is: " & cogoPnt.Latitude
    "The Longitude for the first CogoPoint in the collection is: " & cogoPnt.Longitude,
End Sub
LayerFile Example
Sub Example_LayerFile()
' This example displays the LayerFile setting for the current drawing.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.preferences
Dim layerfile As String
layerfile = dbPref.LayerFile
' Test if layer file is set
If layerfile = "" Then
layerfile = "not set"
End If

MsgBox "The current value for LayerFile is: " & layerfile, vbInformation, "LayerFile Exam
End Sub


LayerStandard Example
Sub Example_LayerStandard()
' This example displays the LayerStandard setting for the current drawing.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.preferences
Dim laystandard As String
laystandard = dbPref.layerstandard
' Test if layer standard is set
If laystandard = "" Then
laystandard = "not set"
End If
MsgBox "The current value for LayerStandard is: " & laystandard, _
vbInformation, "LayerStandard Example"
End Sub


Sub Example_LeftWidth()

  ' This example returns the left width for the
  ' first alignment cross section in the collection.
  Dim alignXsects As AeccCrossSectionBlocks
  Dim alignXsect As AeccCrossSectionBlock
  Set alignXsects = AeccApplication.ActiveDocument.CrossSectionBlocks
  Set alignXsect = alignXsects.Item(0)

  'Get the station for the first alignment cross section in the collection
  Dim station As String
  station = alignXsect.station

  MsgBox "The left width for the alignment cross section at station " & station & 
      Format(alignXsect.LeftWidth, "0.00"), vbInformation, "LeftWidth Example"

End Sub
Length Example

Examples:

1 AlignEntity, AlignCurve, AlignSpiral, AlignTangent
1 ParcelEntity, ParcelCurve, ParcelLine

Sub Example_Length_AlgEntity()

' This example returns the Length for the first entity in the
' first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnt As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set alignEnt = align.AlignEntities.Item(0)

MsgBox "The Length of the first entity in the Alignment is: " & alignEnt.Length,
vbInformation, "Length Example"

End Sub

Sub Example_Length_ParcelEntity()

' This example returns the Length for the first entity in the
' first Parcel in the collection.
Dim parcel As AeccParcel
Dim parcelEnt As AeccParcelEntity
Set parcel = AeccApplication.ActiveProject.Parcels.Item(0)
Set parcelEnt = parcel.ParcelEntities.Item(0)

MsgBox "The Length of the first entity in the Parcel is: " & parcelEnt.Length,
vbInformation, "Length Example"

End Sub
Sub Example_LExt()

' This example returns the Lext value for the first Spiral found in the
' first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Spiral entity in the first Alignment."

' Find first Spiral in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kSpiral Then
        alignMsg = "The Lext value for the first Spiral in the alignment is: " & ali
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "LExt Example"

End Sub
Sub Example_LinearDisplayFormat()

' This example returns the LinearDisplayFormat setting for the current drawing
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.preferences

' Convert the linear display format to a string.
Dim linearformat As String
Select Case dbPref.LinearDisplayFormat
Case 1
   linearformat = "Scientific"
Case 2
   linearformat = "Decimal"
Case 3
   linearformat = "Engineering"
Case 4
   linearformat = "Architectural"
Case 5
   linearformat = "Fractional"
End Select

MsgBox "The current value for LinearDisplayFormat is: " & linearformat, vbInformation
End Sub
Sub Example_LinearPrecision()

' This example returns the LinearPrecision setting for the current drawing.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.Preferences

MsgBox "The current value for LinearPrecision is: " & dbPref.LinearPrecision

End Sub
Sub Example_LinearUnit()

' This example returns the LinearUnit setting for the current drawing.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.preferences

' Convert the volume display unit to a string.
Dim unit As String
Select Case dbPref.LinearUnit
Case aecUnitInch
    unit = "inch"
Case aecUnitFoot
    unit = "foot"
Case aecUnitYards
    unit = "yard"
Case aecUnitMil
    unit = "millimeters"
Case aecUnitCentimeter
    unit = "centimeters"
Case aecUnitDecimeter
    unit = "decimeters"
Case aecUnitMeter
    unit = "meters"
End Select

MsgBox "The current value for LinearUnit is: " & unit, vbInformation, "LinearUnit Example"
End Sub
Sub Example_LineIntersection()

' This example gets the LineIntersection from a user supplied point
' for the first Alignment in the collection..
Dim align As AeccAlignment
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim pnt1 As Variant
Dim pnt2 As Variant
Dim coords As Variant

' Get the points
pnt1 = ThisDrawing.Utility.GetPoint(, "Enter the first point near the first alignment:")
pnt2 = ThisDrawing.Utility.GetPoint(, "Enter the second point near the first alignment:")

' Get the line intersection location
coords = align.LineIntersection(pnt1(0), pnt1(1), pnt2(0), pnt2(1))

MsgBox "The LineIntersection for the first Alignment is:" & vbCrLf & vbCrLf & _
    " Station: " & coords(0) & vbCrLf & vbCrLf & _
    " Direction: " & coords(1) & vbCrLf & vbCrLf & _
    " Easting: " & coords(2) & vbCrLf & vbCrLf & _
    " Northing: " & coords(3), vbInformation, "LineIntersection Example"

End Sub
Sub Example_LoadSetupProfile()

' This example loads an existing drawing setup profile.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.DatabasePreferences

' If no extension is supplied, "set" is appended automatically.
Pref.LoadSetupProfile("i20.set")

End Sub
Sub Example_LockedPointNumbers()

' This examples locks a series of point numbers and
' then unlocks a subset of those point numbers.
Dim cogoPnts As AeccCogoPoints
Set cogoPnts = AeccApplication.ActiveProject.CogoPoints

' Create locked point string
Dim pntString As String
pntString = "1,4-8,40,10,12"

' Lock points
cogoPnts.LockPoints (pntString)

MsgBox "Locked point numbers are " & cogoPnts.LockedPointNumbers, _
vbInformation, "LockedPointNumbers Example"

' Create unlock point string
pntString = "4-8"

' Unlock points
cogoPnts.UnlockPoints (pntString)

MsgBox "Locked point numbers are " & cogoPnts.LockedPointNumbers, _
vbInformation, "LockedPointNumbers Example"

End Sub
Sub Example_LockPoints()

' This examples locks a series of point numbers and
' then unlocks a subset of those point numbers.
Dim cogoPnts As AeccCogoPoints
Set cogoPnts = AeccApplication.ActiveProject.CogoPoints

' Create locked point string
Dim pntString As String
pntString = "1,4-8,40,10,12"

' Lock points
CogoPnts.LockPoints (pntString)

MsgBox "Locked point numbers are " & CogoPnts.LockedPointNumbers, _
    vbInformation, "LockPoints Example"

' Create unlock point string
pntString = "4-8"

' Unlock points
CogoPnts.UnlockPoints (pntString)

MsgBox "Locked point numbers are " & CogoPnts.LockedPointNumbers, _
    vbInformation, "LockPoints Example"

End Sub
Sub Example_LockType()

    ' This example returns the LockType setting for the first Alignment in the collection.
    Dim align As AeccAlignment
    Set align = AeccApplication.ActiveProject.Alignments.Item(0)

    ' Convert the lock type to a string.
    Dim LockType As String
    Select Case align.LockType
        Case kNoLock
            LockType = "no lock exists."
        Case kReadLock
            LockType = "the user can only read the object."
        Case kWriteLock
            LockType = "the user can read and write the object."
    End Select

    MsgBox "The current value for LockType is " & LockType, vbInformation, "LockType Example"

End Sub
Sub Example_LOffset()

    ' This example returns the LOffset for the first Spiral found in the
    ' first Alignment in the collection.
    Dim align As AeccAlignment
    Dim alignEnts As AeccAlignEntity
    Set align = AeccApplication.ActiveProject.Alignments.Item(0)

    Dim alignMsg As String
    alignMsg = "There is no Spiral entity in the first Alignment."

    ' Find first Spiral in the alignment
    For Each alignEnt In align.AlignEntities
        If alignEnt.Type = kSpiral Then
            alignMsg = "The LOffset for the first Spiral in the alignment is: " & alignEnt.LOffset
            Exit For
        End If
    Next

    MsgBox alignMsg, vbInformation, "LOffset Example"

End Sub
Sub Example_Longitude()

' This example returns the Latitude and Longitude
' for the first CogoPoint in the collection.
Dim cogoPnt As AeccCogoPoint
Set cogoPnt = AeccApplication.ActiveProject.CogoPoints.Item(0)

MsgBox "The Latitude for the first CogoPoint in the collection is: " & cogoPnt.Latitude & vbCrLf & "The Longitude for the first CogoPoint in the collection is: " & cogoPnt.Longitude,

End Sub
MaxElevation Example

Examples:

1 CrossSection

1 Surface

Sub Example_MaxElevation_CrossSection()

' This example returns the maximum elevation for the first cross section in the
' cross section collection for the first alignment in the collection.
Dim aligns As AeccAlignments
Dim align As AeccAlignment
Dim xSect As AeccCrossSection
Set aligns = AeccApplication.ActiveProject.Alignments
Set align = aligns.Item(0)
Set xSect = align.CrossSections.Item(0)

' Get the cross section station and format it
Dim station As String
station = aligns.DoubleToStaFormat(xSect.station)

MsgBox "The maximum elevation for the cross section at station " & station &
        Format(xSect.MaxElevation, "0.00"), vbInformation, "MaxElevation Example"

End Sub

Sub Example_MaxElevation_Surface()

' This example returns the MaxElevation for the first surface in the collection.
Dim surf As AeccSurface
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

MsgBox "The MaxElevation for surface " & surf.Name & ", " & Format(surf.MaxElevation, "0.00")
        , vbInformation, "MaxElevation Example"

End Sub
End Sub
Sub Example_MaxFaceArea()

    ' This example returns the MaxFaceArea for the first surface in the collection.
    Dim surf As AeccSurface
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

    MsgBox "The MaxFaceArea for the first surface is: " & Format(surf.MaxFaceArea, "0.00"),
        vbInformation, "MaxFaceArea Example"

End Sub
MagGrade Example

Sub Example_MaxGrade()

    ' This example returns the MaxGrade for the first surface in the collection.
    Dim surf As AeccSurface
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

    MsgBox "The MaxGrade for the first surface is: " & Format(surf.MaxGrade, ",")
    vbInformation, "MaxGrade Example"

End Sub
Sub Example_MaxOffset()

' This example returns the maximum offset for the first cross section in the 
' cross section collection for the first alignment in the collection.
Dim aligns As AeccAlignments
Dim align As AeccAlignment
Dim xSect As AeccCrossSection
Set aligns = AeccApplication.ActiveProject.Alignments
Set align = aligns.Item(0)
Set xSect = align.CrossSections.Item(0)

' Get the cross section station and format it
Dim station As String
station = aligns.DoubleToStaFormat(xSect.station)

MsgBox "The maximum offset for the cross section at station " & station & " is: 
Format(xSect.MaxOffset, "0.00"), vbInformation, "MaxOffset Example"

End Sub
MeanElevation Example

Sub Example_MeanElevation()

    ' This example returns the MeanElevation for the first surface in the collection.
    Dim surf As AeccSurface
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

    MsgBox "The MeanElevation for the first surface is: " & Format(surf.MeanElevation, "0.00"), vbInformation, "MeanElevation Example"

End Sub
**MeasurementUnit Example**

Sub Example_MeasurementUnit()

' This example returns the MeasurementUnit setting for the current drawing.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.preferences

' Convert the measurement unit to a string.
Dim unit As String
If dbPref.MeasurementUnit = acEnglish Then
    unit = "english"
Else
    unit = "metric"
Else
End If

MsgBox "The current value for MeasurementUnit is " & unit, _
    vbInformation, "MeasurementUnit Example"

End Sub
Sub Example_MidOrdinate()

' This example returns the MidOrdinate for the first Curve found in the
' first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Curve entity in the first Alignment."

' Find first Curve in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kCurve Then
        alignMsg = "The MidOrdinate for the first Curve in the alignment is: " &
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "MidOrdinate Example"

End Sub
Sub Example_MinDepressionArea()

    ' This example returns the MinDepressionArea setting for the WaterSheds.
    Dim surf As AeccSurface
    Dim wSheds As AeccWaterSheds
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
    Set wSheds = surf.Outputs.WaterSheds

    MsgBox "The MinDepressionArea setting for WaterSheds is: " & wSheds.MinDepressionArea, vbInformation, "MinDepressionArea Example"

End Sub
Sub Example_MinDepressionDepth()

    ' This example returns the MinDepressionDepth setting for the WaterSheds.
    Dim surf As AeccSurface
    Dim wSheds As AeccWaterSheds
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
    Set wSheds = surf.Outputs.WaterSheds

    MsgBox "The MinDepressionDepth setting for WaterSheds is: " & wSheds.MinDepressionDepth, vbInformation, "MinDepressionDepth Example"

End Sub
MinElevation Example

Examples:

1 CrossSection
1 Surface

Sub Example_MinElevation_CrossSection()

' This example returns the minimum elevation for the first cross section in the
' cross section collection for the first alignment in the collection.
Dim aligns As AeccAlignments
Dim align As AeccAlignment
Dim xSect As AeccCrossSection
Set aligns = AeccApplication.ActiveProject.Alignments
Set align = aligns.Item(0)
Set xSect = align.CrossSections.Item(0)

' Get the cross section station and format it
Dim station As String
station = aligns.DoubleToStaFormat(xSect.station)

MsgBox "The minimum elevation for the cross section at station " & station & 
    Format(xSect.MinElevation, "0.00"), vbInformation, "MinElevation Example"

End Sub

Sub Example_MinElevation_Surface()

' This example returns the MinElevation for the first surface in the collection.
Dim surf As AeccSurface
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

MsgBox "The MinElevation for surface " & surf.Name & 
    " is: " & Format(surf.MinElevation, "0.00"), vbInformation, "MinElevation Example"
End Sub
MinFaceArea Example

Sub Example_MinFaceArea()

    ' This example returns the MinFaceArea for the first surface in the collection.
    Dim surf As AeccSurface
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

    MsgBox "The MinFaceArea for the first surface is: " & Format(surf.MinFaceArea, "0.00"), vbInformation, "MinFaceArea Example"

End Sub
Sub Example_MinGrade()

' This example returns the MinGrade for the first surface in the collection.
Dim surf As AeccSurface
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

MsgBox "The MinGrade for the first surface is " & Format(surf.MinGrade, "0.00\%"), vbInformation, "MinGrade Example"

End Sub
Sub Example_MinOffset()

    ' This example returns the minimum offset for the first cross section in the
    ' cross section collection for the first alignment in the collection.
    Dim aligns As AeccAlignments
    Dim align As AeccAlignment
    Dim xSect As AeccCrossSection
    Set aligns = AeccApplication.ActiveProject.Alignments
    Set align = aligns.Item(0)
    Set xSect = align.CrossSections.Item(0)

    ' Get the cross section station and format it
    Dim station As String
    station = aligns.DoubleToStaFormat(xSect.station)

    MsgBox "The minimum offset for the cross section at station " & station & " is
    Format(xSect.MinOffset, "0.00"), vbInformation, "MinOffset Example"

End Sub
**Modified Example**

**class Module Code:**

```vbnet
Public WithEvents cogoPnts As AeccCogoPoints

Private Sub cogoPnts_Modified()
    MsgBox "Points have been modified", vbInformation, "Modified Example"
End Sub
```

**Module Code:**

```vbnet
Option Explicit
Dim eh As New EventHandler

Sub example_Modify_Event()
    ' This example establishes event handling for the CogoPoints object.
    Set eh.cogoPnts = AeccApplication.ActiveProject.CogoPoints
End Sub

Sub Example_Modify()
    ' This example change toggles the raw description of the first point in the collection to illustrate the Modify event.
    Dim cogoPnt As AeccCogoPoint
    Set cogoPnt = AeccApplication.ActiveProject.CogoPoints.Item(0)

    ' Get the points raw description
    Dim strName As String
    strName = cogoPnt.RawDescription

    ' Modify the raw description
```
If cogoPnt.RawDescription = "Old Description" Then
    cogoPnt.RawDescription = "New Description"
Else
    cogoPnt.RawDescription = "Old Description"
End If

' Show the override description on the command line
ThisDrawing.Utility.Prompt vbCrLf & "The raw description of the first point is:"
Examples:

- AeccContourStyle
- Alignment
- CogoPoint
- CrossSectionBlock (Civil Engineering Feature)
- CrossSectionSurface (Civil Engineering Feature)
- DEMFile
- DescriptionKeyFile
- Drawing
- FGProfiles (Civil Engineering Feature)
- Parcel
- PointFile
- PointGroup
- PointGroupName
- ProfileBlock (Civil Engineering Feature)
- Project
- Prototype
- Surface

Sub Example_Name_AeccContourStyle()
' This example returns the Name for the contour style in the dictionary
Dim objContourStyle As AeccContourStyle
Set objContourStyle = ThisDrawing.Dictionaries("AECC_CONTOUR_STYLES").Item(0)
MsgBox "The value for the ContourStyle Name is: " & objContourStyle.Name,
End Sub

Sub Example_Name_Alignment()

' This example returns the Name for the first Alignment in the collection
Dim align As AeccAlignment
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
MsgBox "The Name of the first Alignment is: " & align.Name,
End Sub

Sub Example_Name_CogoPoint()

' This example returns the Name for the first CogoPoint in the collection.
Dim cogoPnt As AeccCogoPoint
Set cogoPnt = AeccApplication.ActiveProject.CogoPoints.Item(0)
MsgBox "The Name for the first CogoPoint in the collection is: " & cogoPnt.Name,
End Sub

Sub Example_Name_CrossSectionBlock()

' This example returns the name of the profile for the
' first alignment cross section in the collection.
Dim alignXsects As AeccCrossSectionBlocks
Dim alignXSect As AeccCrossSectionBlock
Set alignXsects = AeccApplication.ActiveDocument.CrossSectionBlocks
Set alignXSect = alignXsects.Item(0)
'Get the station for the first alignment cross section in the collection
Dim station As String
station = alignXsect.station

MsgBox "The profile name for the alignment cross section at station " & station & 
alignXsect.Name, vbInformation, "Name Example"

End Sub

Sub Example_Name_CrossSectionSurface()

' This example returns the name for the first cross section surface in the collection for the first cross section in the collection.
Dim aligns As AeccAlignments
Dim align As AeccAlignment
Dim xSect As AeccCrossSection
Dim xSectSurf As AeccCrossSectionSurface
Set aligns = AeccApplication.ActiveProject.Alignments
Set align = aligns.Item(0)
Set xSect = align.CrossSections.Item(0)
Set xSectSurf = xSect.CrossSectionSurfaces.Item(0)

' Get the cross section station and format it
Dim station As String
station = aligns.DoubleToStaFormat(xSect.station)

MsgBox "The name for the first cross section surface for " & vbCrLf & _
"the cross section at station " & station & " is: " & _
xSectSurf.Name, vbInformation, "Name Example"

End Sub

Sub Example_Name_DEMFile()

' This example returns the Name for the first DEMFile in the first surface in the collection.
Dim surf As AeccSurface
Dim DEMFile As AeccDEMFile
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set DEMFile = surf.Inputs.DEMFiles.Item(0)

MsgBox "The Name for the first DEMFile is: " & DEMFile.Name, _
    vbInformation, "Name Example"

End Sub

Sub Example_Name_DescriptionKeyFile()

    ' This function gets the DescriptionKeyFile name
    Dim dKeyFile As AeccDescriptionKeyFile
    Set dKeyFile = AeccApplication.ActiveProject.DescriptionKeyFiles.Item(0)

    MsgBox "The Name for the first DescriptionKeyFile in the collection is: " & dKeyFile.Name, _
        vbInformation, "Name Example"

End Sub

Sub Example_Name_Drawing()

    ' This example returns the Name setting for the first Drawing
    ' in the collection
    Dim dwg As AeccDrawing
    Set dwg = AeccApplication.ActiveProject.Drawings(0)

    MsgBox "The Name for the first Drawing in the collection is: " & dwg.Name, _
        vbInformation, "Name Example"

End Sub

Sub Example_Name_FGProfiles()

    ' This example returns the Name setting of the FGProfile
    ' for the first alignment in the collection
Dim align As AeccAlignment
Dim FGProfs As AeccFGProfiles
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set FGProfs = align.FGProfiles

MsgBox "The Name of the finished ground profile for the first Alignment is: " & 
    FGProfs.Name, vbInformation, "Name Example"

End Sub

Sub Example_Name_Parcel()
    ' This example starts by creating a Parcel named "NewParcel".
    ' The new Parcel name displayed. Finally, the new Parcel is renamed
    ' to "OldName" and is displayed again.
    Dim parcels As AeccParcels
    Dim parcel As AeccParcel
    Set parcels = AeccApplication.ActiveProject.Parcels

    ' ' Add a new Parcel name "NewParcel"
    Set parcel = parcels.Add("NewParcel")

    MsgBox "The Parcel name is: " & parcel.Name, vbInformation, "Name Example"

    ' ' Rename the new Parcel to "OldParcel"
    parcels.Rename "NewParcel", "OldParcel"

    MsgBox "The Parcel name is: " & parcel.Name, vbInformation, "Name Example"

End Sub

Sub Example_Name_PointFile()
    ' This example returns the Name setting for the first PointFile
    ' in the collection.
    Dim surf As AeccSurface
    Dim pntFile As AeccPointFile
Sub Example_Name_PointGroup()

' This function gets the Name for the first PointGroup 
' in the collection.
Dim pntGrp As AeccPointGroup
Set pntGrp = AeccApplication.ActiveProject.PointGroups.Item(0)

MsgBox "The Name for the first PointGroup is: " & _
pntGrp.Name, vbInformation, "Name Example"

End Sub

Sub Example_Name_PointGroupName()

' This example returns the Name setting for the first PointGroupName 
' in the collection.
Dim surf As AeccSurface
Dim pntGrpName As AeccPointGroupName
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set pntGrpName = surf.Inputs.PointGroupNames.Item(0)

MsgBox "The Name for the first PointGroupName in the collection is: " & _
pntGrpName.Name, vbInformation, "Name Example"

End Sub

Sub Example_Name_ProfileBlock()

' This example returns the Name for the first ProfileBlock in the collection
Dim alignProf As AeccProfileBlock
Set alignProf = AeccApplication.ActiveDocument.ProfileBlocks.Item(0)

MsgBox "The Name of the first ProfileBlock in the collection is: " & alignProf.Name, vbInformation, "Name Example"

End Sub

Sub Example_Name_Project()

' This example returns the Name setting for the first Project in the collection
Dim proj As AeccProject
Set proj = AeccApplication.Projects.Item(0)

MsgBox "The Name value for the first Project in the collection is: " & proj.Name, vbInformation, "Name Example"

End Sub

Sub Example_Name_Prototype()

' This example returns the Name of the first prototype in the collection.
Dim prot As AeccPrototype
Set prot = AeccApplication.Prototypes.Item(0)

MsgBox "The Name of the first prototype in the collection is: " & prot.Name, vbInformation, "Name Example"

End Sub

Sub Example_Name_Surface()

' This example returns the Name of the first surface in the collection.
Dim surf As AeccSurface
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

MsgBox "The Name of the first surface is: " & surf.Name, vbInformation, "Name Example"
End Sub
Sub Example_NameOverride()

    ' This function gets the NameOverride for the first PointGroup
    ' in the collection.
    Dim pntGrp As AeccPointGroup
    Set pntGrp = AeccApplication.ActiveProject.PointGroups.Item(0)

    If pntGrp.NameOverride = True Then
        MsgBox "The NameOverride for the first PointGroup is turned on.", _
        vbInformation, "NameOverride Example"
    Else
        MsgBox "The NameOverride for the first PointGroup is turned off.", _
        vbInformation, "NameOverride Example"
    End If

End Sub
Sub Example_NameXDRef()

    ' This function gets the Point group point name XDRef
    Dim ptGrp As AeccPointGroup
    Set ptGrp = AeccApplication.ActiveProject.PointGroups.Item(0)

    MsgBox "The point group point NameXDRef is: " & ptGrp.NameXDRef, _
        vbInformation, "NameXDRef Example"

End Sub
Sub Example_NewProjectBased()

' This example creates a new project-based drawing.
Dim doc As AeccDocument
Dim newDoc As AeccDocument
Set doc = AeccApplication.ActiveDocument

' Specify the full path of the new drawing
Dim strNewDwg As String
strNewDwg = doc.Path + "\New Drawing"

' Specify the full path of the template
Dim strTemplate As String
strTemplate = AeccApplication.Application.Preferences.Files.TemplateDwgPath

Set newDoc = doc.NewProjectBased(strTemplate, strNewDwg)

MsgBox "The new drawing is named:" & newDoc.Name, vbInformation, "NewProjectBased Example"

End Sub
Sub Example_NextPointNumber()

    ' This gets the NextPointNumber in the CogoPoints collection
    Dim cogoPnts As AeccoCogoPoints
    Set cogoPnts = AeccoApplication.ActiveProject.CogoPoints

    MsgBox "The NextPointNumber for CogoPoints is: " & cogoPnts.NextPointNumber,
            vbInformation, "NextPointNumber Example"

End Sub
Normal Example

Sub Example_Normal()

' This example returns the normal value for the first face item in the collection.
Dim surf As AeccSurface
Dim face As AeccFace
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set face = surf.Outputs.Faces.Item(0)

Dim norm As Variant

norm = face.Normal

MsgBox "The Normal value for the first Face is " & norm(0) & ", " & norm(1) 
vbInformation, "Normal Example"

End Sub
Northing Example

Examples:

- CommPoint
- CogoPoint
- TinPoint

Sub Example_Northing_AeccPoint()

' This example returns the Northing setting for the first Point object in a selection set

On Error Resume Next

' Delete existing SelectionSet
ThisDrawing.SelectionSets("SSet").Delete

' Create the selection set based on a point selection and filter for the Point objects
Dim ssetObj As AcadSelectionSet
Set ssetObj = ThisDrawing.SelectionSets.Add("SSet")

Dim gpCode(0) As Integer
Dim dataValue(0) As Variant

gpCode(0) = 0
dataValue(0) = "AECC_POINT"

Dim groupCode As Variant
Dim dataCode As Variant

groupCode = gpCode
dataCode = dataValue
Dim objPoint As AeccPoint
Set objPoint = ssetObj.Item(0)

MsgBox "The setting for Northing is: " & objPoint.Northing, vbInformation, "Northing Example"
End Sub

Sub Example_Northing_CogoPoint()

' This example returns the Northing for the first CogoPoint in the collection
Dim cogoPnt As AeccCogoPoint
Set cogoPnt = AeccApplication.ActiveProject.CogoPoints.Item(0)

MsgBox "The Northing for the first CogoPoint in the collection is: " & cogoPnt.Northing, vbInformation, "Northing Example"
End Sub

Sub Example_Northing_TinPoint()

' This example returns the Northing of the first
' TIN Point in the collection.
Dim surf As AeccSurface
Dim tin As AeccTinPoint
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set tin = surf.Outputs.TinPoints.Item(0)

MsgBox "The Northing of the first TIN Point is: " & tin.Northing, vbInformation, "Northing Example"
End Sub
Sub Example_NorthRotation()

' This example returns the NorthRotation setting for the current drawing.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.Preferences

MsgBox "The current value for NorthRotation is: " & dbPref.NorthRotation, _
    vbInformation, "NorthRotation Example"

End Sub
**Number Example**

**Examples:**

- [AeccPoint](#)
- [Alignment](#)
- [CogoPoint](#)
- [Parcel](#)

---

```vba
Sub Example_Number_AeccPoint()

    ' This example returns the Number setting for the first Point object in a selection set
    On Error Resume Next

    ' Delete existing SelectionSet
    ThisDrawing.SelectionSets("SSet").Delete

    ' Create the selection set based on a point selection and filter for the Point objects
    Dim ssetObj As AcadSelectionSet
    Set ssetObj = ThisDrawing.SelectionSets.Add("SSet")

    Dim gpCode(0) As Integer
    Dim dataValue(0) As Variant

    gpCode(0) = 0
    dataValue(0) = "AECC_POINT"

    Dim groupCode As Variant
    Dim dataCode As Variant

    groupCode = gpCode
```
dataCode = dataValue
ssetObj.SelectOnScreen groupCode, dataCode

Dim objPoint As AeccPoint
Set objPoint = ssetObj.Item(0)

MsgBox "The setting for Number is: " & objPoint.Number, vbInformation, "Number Example"

End Sub

Sub Example_Number_Alignment()

' This example returns the Number for the first Alignment in the collection.
Dim align As AeccAlignment
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

MsgBox "The Number of the first alignment is: " & align.Number, vbInformation

End Sub

Sub Example_Number_CogoPoint()

' This example returns the Number for the first CogoPoint in the collection.
Dim cogoPnt As AeccCogoPoint
Set cogoPnt = AeccApplication.ActiveProject.CogoPoints.Item(0)

MsgBox "The Number for the first CogoPoint in the collection is: " & cogoPnt.Number, vbInformation

End Sub

Sub Example_Number_Parcel()

' This example returns the parcel Number for the first parcel in the collection.
Dim parcel As AeccParcel
Set parcel = AeccApplication.ActiveProject.parcels.Item(0)

MsgBox "The Number for the first Parcel in the collection is: "

& parcel.Number, vbInformation, "Number Example"

End Sub
Sub Example_NumberOfFaces()

' This example returns the NumberOfFaces for the first surface in the collection
Dim surf As AeccSurface
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

MsgBox "The NumberOfFaces for the first surface is: " & surf.NumberOfFace
    vbInformation, "NumberOfFaces Example"

End Sub
NumberOfPoints Example

Sub Example_NumberOfPoints()

    ' This example returns the NumberOfPoints for the first surface in the collection
    Dim surf As AeccSurface
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

    MsgBox "The NumberOfPoints for the first surface is: " & surf.NumberOfPoints

End Sub
Sub Example_ObjectID()

' This example returns the ObjectId value for the first Tangent found in the first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Tangent entity in the first Alignment."

' Find first Tangent in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kTangent Then
        alignMsg = "The ObjectId for the first Tangent in the alignment is: " & alignEnt.ObjectID
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "ObjectId Example"

End Sub
Sub Example_Offset()

' This example returns the point code data for the first cross
' section surface in the collection for the first cross section
' in the collection.
Dim aligns As AeccAlignments
Dim align As AeccAlignment
Dim xSect As AeccCrossSection
Dim xSectPCode As AeccCrossSectionPointCode
Set aligns = AeccApplication.ActiveProject.Alignments
Set align = aligns.Item(0)
Set xSect = align.CrossSections.Item(0)
Set xSectPCode = xSect.CrossSectionPointCodes.Item(0)

' Get the alignment name
Dim alignName As String
alignName = align.Name

' Get the cross section station and format it
Dim station As String
station = aligns.DoubleToStaFormat(xSect.station)

MsgBox "The alignment name is: " & alignName & vbCrLf & ""The first cross section is at station: " & station & vbCrLf & l"
"The data for the first point code is: " & vbCrLf & l"
vbTab & "Code: " & xSectPCode.Code & vbCrLf & l"
vbTab & "Description: " & xSectPCode.Description & vbCrLf & l"
vbTab & "Elevation: " & Format(xSectPCode.elevation, "0.00") & vbCrLf & l"
vbTab & "Offset: " & Format(xSectPCode.offset, "0.00"), _, vbCrLf & "Offset Example"

End Sub
Sub Example_OffsetElevations()

' This example returns the first three offsets and elevations for the first cross
' section surface in the collection for the first cross section
' in the collection.
Dim aligns As AeccAlignments
Dim align As AeccAlignment
Dim xSect As AeccCrossSection
Dim xSectSurf As AeccCrossSectionSurface
Set aligns = AeccApplication.ActiveProject.Alignments
Set align = aligns.Item(0)
Set xSect = align.CrossSections.Item(0)
Set xSectSurf = xSect.CrossSectionSurfaces.Item(0)

' Get the alignment name
Dim alignName As String
alignName = align.Name

' Get the cross section station and format it
Dim station As String
station = aligns.DoubleToStaFormat(xSect.station)

' Get the cross section surface name
Dim surfName As String
surfName = xSectSurf.Name

' Get the offsets and elevations
Dim offElev As Variant
offElev = xSectSurf.OffsetElevations

MsgBox "The alignment name is: " & alignName & vbCrLf & _
"The first cross section is at station: " & station & vbCrLf & _
"The first surface name is: " & surfName & vbCrLf & _
"The first three offsets and elevations for the surface are:" & vbCrLf & _
vbTab & "Offset 1: " & Format(offElev(0), "0.00") & _
vbTab & "Elevation 1: " & Format(offElev(1), "0.00") & vbCrLf & _
vbTab & "Offset 2: " & Format(offElev(2), "0.00") & _
vbTab & "Elevation 2: " & Format(offElev(3), "0.00") & vbCrLf & _
vbTab & "Offset 3: " & Format(offElev(4), "0.00") & _
vbTab & "Elevation 3: " & Format(offElev(5), "0.00"), _
End Sub
Sub Example_OffsetElevationToXy()

' This example returns an Autocad XY for a given offset and elevation for the
' first alignment cross section in the collection.
Dim alignXSect As AeccCrossSectionBlock
Set alignXSect = AeccApplication.ActiveDocument.CrossSectionBlocks.Item(0)

Dim offElev(0 To 1) As Double
Dim XY As Variant
Dim station As String
Dim offset As Double
Dim elevation As Double

' Get the station for the first alignment cross section in the collection.
station = alignXSect.station

' Get the offset and the elevation
offset = ThisDrawing.Utility.GetReal(vbCrLf & "Enter the offset in station " & station & ":");
elevation = ThisDrawing.Utility.GetReal("Enter the elevation in station " & station & ":")

' Use the entered values to get the X and Y values
offElev(0) = offset
offElev(1) = elevation

XY = alignXSect.OffsetElevationToXy(offElev)

MsgBox "The X value for the offset is: " & Format(XY(0), "0.00") & vbCrLf & 
"The Y value for the elevation is: " & Format(XY(1), "0.00") _
, vbInformation, "OffsetElevationToXy Example"

End Sub
Sub Example_Open()

    ' This example opens an existing project-based drawing.
    Dim dwgs As AeccDrawings
    Set dwgs = AeccApplication.ActiveProject.Drawings

    ' Specify the full path of the drawing
    Dim strDwg As String
    Set strDwg = AeccApplication.ActiveProject.Drawings.Item(0).FullName

    ' Specify the full path of the project
    Dim strProject As String
    Set strProject = AeccApplication.ActiveProject.FullName

    dwgs.Open strDwg, strProject

End Sub
Sub Example_OpenProjectBased()

' This example opens the first existing project-based drawing
' in the collection.
Dim dwg As AeccDrawing
Dim doc As AeccDocument
Dim openDoc As AeccDocument

Set dwg = AeccApplication.ActiveProject.Drawings.Item(0)
Set doc = AeccApplication.ActiveDocument

' Specify the full path of the drawing
Dim strDwg As String
strDwg = dwg.FullName

Set openDoc = doc.OpenProjectBased(strDwg)

MsgBox "The name of the opened drawing is " & openDoc.Name, _
    vbCrLfInformation, "OpenProjectBased Example"

End Sub
Sub Example_Output()

' This example returns the Count of Faces for the first Surface
' in the collection by using the Outputs property.
Dim surf As AeccSurface
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

MsgBox "The number of Faces in the first surface is: " & surf.Outputs.Faces.Count, vbInformation, "Outputs Example"

End Sub
OverflowPoints Example

Sub Example_OverflowPoints()

' This example returns the first point in the OverflowPoints
' for the first WaterShed in the collection.
Dim surf As AeccSurface
Dim wShed As AeccWaterShed
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set wShed = surf.Outputs.WaterSheds.Item(0)

Dim coords As Variant

coords = wShed.OverflowPoints

MsgBox "The first OverflowPoint for the first WaterShed is: " & coords(0) &
, vbInformation, "OverflowPoints Example"

End Sub
OverrideDescription Example

Sub Example_OverrideDescription()

    ' This example returns the OverrideDescription for the first CogoPoint in the collection.
    ' This example the first point in the collection is in a Point Group Named "Example Group".
    Dim cogoPnts As AeccCogoPoints
    Dim cogoPnt As AeccCogoPoint
    Set cogoPnts = AeccApplication.ActiveProject.CogoPoints

    ' Filter CogoPoints collection on GroupName
    cogoPnts.GroupName = "Example Group"

    Set cogoPnt = cogoPnts.Item(0)

    MsgBox "The OverrideDescription for the first CogoPoint in the collection is:"
        cogoPnt.OverrideDescription, vbInformation, "OverrideDescription Example"

End Sub
OverrideElevation Example

Sub Example OverrideElevation()

' This example returns the OverrideElevation for the first CogoPoint in the coll
' This example assumes the first point in the collection is in a
' Point Group named "Example Group".
Dim cogoPnts As AeccCogoPoints
Dim cogoPnt As AeccCogoPoint
Set cogoPnts = AeccApplication.ActiveProject.CogoPoints

' Filter CogoPoints collection on GroupName
cogoPnts.GroupName = "Example Group"

Set cogoPnt = cogoPnts.Item(0)

MsgBox "The OverrideElevation for the first CogoPoint in the collection is: " & cogoPnt.OverrideElevation, vbInformation, "OverrideElevation Example"

End Sub
OverrideName Example

Sub Example_OVERRIDE_NAME()

' This example returns the OverrideName for the first CogoPoint in the collection.
' This example assumes the first point in the collection is in a Point Group named "Example Group".
Dim cogoPnts As AeCCogoPoints
Dim cogoPnt As AeCCogoPoint
Set cogoPnts = AeCCApplication.ActiveProject.CogoPoints

' Filter CogoPoints collection on GroupName
cogoPnts.GroupName = "Example Group"

Set cogoPnt = cogoPnts.Item(0)

MsgBox "The OverrideName for the first CogoPoint in the collection is: " & _
cogoPnt.OverrideName, vbInformation, "OverrideName Example"

End Sub
OverrideNew Example

Sub Example_OverrideNew()

' This example returns the OverrideNew setting.
Dim prefUser As AeccPreferencesUser
Set prefUser = AeccApplication.Preferences.User

MsgBox "The current value for OverrideNew is: " & prefUser.OverrideNew, _
   vbInformation, "OverrideNew Example"

End Sub
OverrideOpen Example

Sub Example_OverrideOpen()
    ' This example returns the OverrideOpen setting.
    Dim prefUser As AeccPreferencesUser
    Set prefUser = AeccApplication.Preferences.User

    MsgBox "The current value for OverrideOpen is: " & prefUser.OverrideOpen, vbInformation, "OverrideOpen Example"
End Sub
Sub Example_Owner

' This example returns the Label for the first FileLock in the collection.
Dim filelock As AeccFileLock
Set filelock = AeccApplication.ActiveProject.fileLocks.Item(0)

MsgBox "The Label of the first FileLock in the collection is: " & filelock.Label
, vbInformation, "Label Example"

End Sub
Sub Example_P()

' This example returns the P value for the first Spiral found in the first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Spiral entity in the first Alignment."

' Find first Spiral in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kSpiral Then
        alignMsg = "The P value for the first Spiral in the alignment is: " & alignEnt.P
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "P Example"

End Sub
Sub Example_Parcel()

' This example returns the K Label layer for Parcel Preferences
' in the current project.
Dim prefPrj As AeccPreferencesProject
Set prefPrj = AeccApplication.ActiveProject.Preferences

MsgBox "The K Label layer for Parcel Preferences in the current Project is: " _
& prefPrj.Parcel.GetString(kLabelLayer), vbInformation, "Parcel Example"

End Sub
Sub Example_ParcelEntities()

' This example uses the ParcelEntities property to get the number of entities for
' of entities for the first parcel in the collection.
Dim parcel As AeccParcel
Set parcel = AeccApplication.ActiveProject.parcels.Item(0)

MsgBox "The number of entities for the first Parcel in the collection is: " _
& parcel.ParcelEntities.Count, vbInformation, "ParcelEntities Example"

End Sub
Sub Example_Paste()

' This example pastes the second surface in the collection to
' the first surface in the collection
Dim surf As AeccSurface
Dim surfs As AeccSurfaces
Set surfs = AeccApplication.ActiveProject.Surfaces

' Get the name of the second surface in the collection
Set surf = surfs.Item(1)
Dim surfName As String
surfName = surf.Name

' Set the current surface to the first surface in the collection
Set surf = surfs.Item(0)
surfs.CurrentSurface = surf.Name

' Paste the two surfaces
surf.Paste surfName

MsgBox "The current surface name is: " & surfs.CurrentSurface & vbCrLf & "The pasted surface name is: " & surfName, vbInformation, "Paste Example"

End Sub
Path Example

Examples:

- DescriptionKeyFile
- DescriptionKeyFiles
- Drawing
- Drawings
- FileLocks
- Project

Sub Example_Path_DescriptionKeyFile()

' This example returns the Path for the first DescriptionKeyFile in the collection.
Dim dKeyFile As AeccDescriptionKeyFile
Set dKeyFile = AeccApplication.ActiveProject.DescriptionKeyFiles.Item(0)

MsgBox "The value for DescriptionKeyFile Path is: " & dKeyFile.Path, vbInformation, "Path Example"

End Sub

Sub Example_Path_DescriptionKeyFiles()

' This example returns the Path for DescriptionKeyFiles.
Dim dKeyFiles As AeccDescriptionKeyFiles
Set dKeyFiles = AeccApplication.ActiveProject.DescriptionKeyFiles

MsgBox "The value for DescriptionKeyFiles Path is: " & dKeyFiles.Path, vbInformation, "Path Example"

End Sub
Sub Example_Path_Drawing()

' This example returns the Path setting for the first Drawing
' in the collection.
Dim dwg As AeccDrawing
Set dwg = AeccApplication.ActiveProject.Drawings(0)

MsgBox "The Path for the first Drawing in the collection is: " & dwg.Path_, vbInformation, "Path Example"

End Sub

Sub Example_Path_Drawings()

' This example returns the Path for the drawings in the active project.
Dim dwgs As AeccDrawings
Set dwgs = AeccApplication.ActiveProject.Drawings

MsgBox "The Path value for Drawings is: " & dwgs.Path, vbInformation, "Path Example"

End Sub

Sub Example_Path_FileLocks()

' This example returns the Path for the FileLocks in the active project.
Dim fileLocks As AeccFileLocks
Set fileLocks = AeccApplication.ActiveProject.FileLocks

MsgBox "The Path value for FileLocks is: " & fileLocks.Path, vbInformation, "Path Example"

End Sub

Sub Example_Path_Project()

' This example returns the Path setting for the first Project
' in the collection
Dim proj As AeccProject
Set proj = AeccApplication.Projects.Item(0)

MsgBox "The Path value for the first Project in the collection is: " & proj.Path,

End Sub
Sub Example_Perimeter()

    ' This example returns the Perimeter for the first parcel in the collection.
    Dim parcel As AeccParcel
    Set parcel = AeccApplication.ActiveProject.parcels.Item(0)

    MsgBox "The Perimeter for the first Parcel in the collection is: " _
        & Format(parcel.Perimeter, "0.00"), vbInformation, "Perimeter Example"

End Sub
Sub Example_PerpIntersection()

' This example gets the PerpIntersection from a user supplied point ' for the first Alignment in the collection..
Dim align As AeccAlignment
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim returnPnt As Variant
Dim coords As Variant

' Get the point
returnPnt = ThisDrawing.Utility.GetPoint(, "Enter a point near the first alignment:

' Get the perpendicular intersection location
coords = align.PerpIntersection(returnPnt(0), returnPnt(1))

MsgBox "The PerpIntersection for the first Alignment is:" & vbCrLf & _
    " Station: " & coords(0) & vbCrLf & _
    " Direction: " & coords(1) & vbCrLf & _
    " Easting: " & coords(2) & vbCrLf & _
    " Northing: " & coords(3), vbInformation, "PerpIntersection Example"

End Sub
Sub Example_PiEastings()

' This example returns the PiEastings for the first Curve found in the first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Curve entity in the first Alignment."

' Find first Curve in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kCurve Then
        alignMsg = "The OiEastings for the first Curve in the alignment is: " & alignEnt.PiEastings
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "PiEastings Example"

End Sub
Sub Example_PiNorthing()

' This example returns the PiNorthing for the first Curve found in the ' first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Curve entity in the first Alignment."

' Find first Curve in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kCurve Then
        alignMsg = "The PiNorthing for the first Curve in the alignment is: " & alignEnt.PiNorthing
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "PiNorthing Example"

End Sub
Sub Example_PointByNumber()

    ' This gets a CogoPoint given a point number.
    Dim cogoPnts As AeccCogoPoints
    Dim cogoPnt As AeccCogoPoint
    Dim pntNum As Long

    ' Set point number to get to 10
    pntNum = 10
    Set cogoPnts = AeccApplication.ActiveProject.CogoPoints
    Set cogoPnt = cogoPnts.PointByNumber(pntNum)

    MsgBox "The Number for CogoPoints is: " & cogoPnt.Number, vbInformation

End Sub
Sub Example_PointCodeDescription()

' This example returns the point code description for the first cross section in the collection.
Dim align As AeccAlignment
Dim xSects As AeccCrossSections
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set xSects = align.CrossSections

' Get the the description for point code 1
Dim desc As String
desc = xSects.PointCodeDescription(1)

MsgBox "The description for point code 1 is: " & _
"Description: " & desc, vbInformation, "PointCodeDescription Example"

End Sub
PointFiles Example

Sub Example_PointFiles()

' This example returns the number of PointFiles in the first ' surface in the collection.
Dim surf As AeccSurface
Dim surfIn As AeccSurfaceInputs
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set surfIn = surf.Inputs

MsgBox "The number of PointFiles in the first Surface is: " & surfIn.PointFiles.Count , vbInformation, "PointFiles Example"

End Sub
PointGroupNames Example

Sub Example_PointGroupNames()

' This example returns the number of PointGroupNames in the first
' surface in the collection.
Dim surf As AeccSurface
Dim surfIn As AeccSurfaceInputs
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set surfIn = surf.Inputs

MsgBox "The number of PointGroupNames in the first Surface is: " & surfIn.PointGroupNames.Count, vbInformation, "PointGroupNames Example"

End Sub
PointGroups Example

Sub Example_PointGroups()

' This example returns the number of PointGroups
' in the current project.
Dim proj As AeccProject
Set proj = AeccApplication.ActiveProject

MsgBox "The number of PointGroups in the current Project is: " & proj.PointGroups.Count, vbInformation, "PointGroups Example"

End Sub
Sub Example_PointList()

' This function gets the PointList for the first PointGroup
' in the collection.
Dim pntGrp As AeccPointGroup
Set pntGrp = AeccApplication.ActiveProject.PointGroups.Item(0)

MsgBox "The PointList for the first PointGroup is: " & pntGrp.PointList, _
    vbInformation, "PointList Example"

End Sub
Sub Example_PointLocation()

' This example gets the PointLocation from user supplied values
' for the first Alignment in the collection.
Dim align As AeccAlignment
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim station As Double
Dim offset As Double
Dim easting As Double
Dim northing As Double
Dim direction As Double

' Get the station and the offset
station = ThisDrawing.Utility.GetReal("Enter the station for the point: ")
offset = ThisDrawing.Utility.GetReal("Enter the offset for the point: ")

' Get the point location
align.PointLocation station, offset, easting, northing, direction

MsgBox "The Point Location is:" & vbCrLf & _
" Easting: " & easting & vbCrLf & _
" Northing: " & northing & vbCrLf & _
" Direction: " & direction, vbInformation, "PointLocation Example"

End Sub
PointNameSize Example

Sub Example_PointNameSize()

    ' This gets the PointNameSize for the CogoPoints collection
    Dim cogoPnts As AeccCogoPoints
    Set cogoPnts = AeccApplication.ActiveProject.CogoPoints

    MsgBox "The PointNameSize for CogoPoints is: " & cogoPnts.PointNameSize
        vbInformation, "PointNameSize Example"

End Sub
Sub Example_PointNumberSelect()

' This gets all the CogoPoint numbers in a selection set and
' displays them as a comma delimited string.
Dim cogoPnts As AeccCogoPoints
Dim entity As Object
Dim pntNum As Long
Dim odjId As Long
Dim firstPnt As Boolean
Dim strPnts As String

Set cogoPnts = AeccApplication.ActiveProject.CogoPoints

On Error Resume Next

' Delete existing SelectionSet
ThisDrawing.SelectionSets("SSet").Delete

' Create the selection filtered for CogoPoint objects
Dim ssetObj As AcadSelectionSet
Set ssetObj = ThisDrawing.SelectionSets.Add("SSet")

Dim gpCode(0) As Integer
Dim dataValue(0) As Variant

gpCode(0) = 0
dataValue(0) = "AECC_POINT"

Dim groupCode As Variant
Dim dataCode As Variant

groupCode = gpCode
dataCode = dataValue
ssetObj.SelectOnScreen groupCode, dataCode
firstPnt = True
' Process selection set
For Each entity In ssetObj
    objId = entity.ObjectID
    pntNum = cogoPnts.PointNumberFromObjID(objId)
    If pntNum > 0 Then
        If firstPnt = True Then
            strPnts = Format(pntNum)
            firstPnt = False
        Else
            strPnts = strPnts +"," + Format(pntNum)
        End If
    End If
Next

MsgBox "The CogoPoint numbers in the selection are: " & _
    strPnts, vbInformation, "PointNumberFromObjID Example"

End Sub
Sub Example_PointOnLineTolerance()

' This example returns the PointOnLineTolerance for the first surface in the col
Dim surf As AeccSurface
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

MsgBox "The PointOnLineTolerance for the first surface is: " & surf.PointOnL
vbInformation, "PointOnLineTolerance Example"

End Sub
Sub Example_PointStringToArray()

' This examples returns a array of point numbers from a CogoPoints string format.
Dim cogoPnts As AeccCogoPoints
Set cogoPnts = AeccApplication.ActiveProject.CogoPoints

Dim pntString As String
Dim pntArray As Variant

pntString = "1,4-8,40,10,12"

' Convert to array
pntArray = cogoPnts.PointStringToArray(pntString)

MsgBox "The CogoPoints point number for 1,4-8,40,10,12 are " & _
pntArray(0) & " " & pntArray(1) & " " & pntArray(2) & " " & _
pntArray(3) & " " & pntArray(4) & " " & pntArray(5) & " " & _
pntArray(6) & " " & pntArray(7) & " " & pntArray(8), vbInformation, "PointStringToArray Example"

End Sub
Sub Example_PointTolerance()

    ' This example returns the PointTolerance for the first surface in the collection.
    Dim surf As AeccSurface
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

    MsgBox "The PointTolerance for the first surface is: " & surf.PointTolerance, _
      vbInformation, "PointTolerance Example"

End Sub
Sub Example_Precision()

' This example returns the Precision for the first parcel in the collection.
Dim parcel As AeccParcel
Set parcel = AeccApplication.ActiveProject.parcels.Item(0)

MsgBox "The Precision for the first Parcel in the collection is: " _
& parcel.Precision, vbInformation, "Precision Example"

End Sub
Preferences Example

Examples:

1 AeccApplication

1 Document

1 Project

Sub Example_Preferences_AeccApplication()

' This example returns the current setting of PrototypePath
' from the preferences object.
 Dim preferences As AeccPreferences
 Set preferences = AeccApplication.Preferences

MsgBox "The current value for PrototypePath is: " & preferences.Files.PrototypePath,
 vbInformation, "Preferences Example"

End Sub

Sub Example_Preferences_Document()

' This example returns the AngularPrecision for the active document.
 Dim doc As AeccDocument
 Set doc = AeccApplication.ActiveDocument

MsgBox "The setting for AngularPrecision is: " & doc.Preferences.AngularPrecision,
 vbInformation, "Preferences Example"

End Sub

Sub Example_Preferences_Project()

' This example returns the text size for Cogo Preferences
' in the current project.

Dim proj As AeccProject

Set proj = AeccApplication.ActiveProject

MsgBox "The text size for Cogo Preferences in the current Project is: " & _
    proj.Preferences.Cogo.GetDouble(kPntTextSize), vbInformation, "Preferenc

End Sub
PreferencesPath Example

Sub Example_PreferencesPath()

' This example returns the PreferencesPath setting.
Dim prefFiles As AeccPreferencesFiles
Set prefFiles = AeccApplication.Preferences.Files

MsgBox "The current value for PreferencesPath is: " & prefFiles.PreferencesPath, vbInformation, "PreferencesPath Example"

End Sub
Profile Example

Sub Example_Profile()

    ' This example returns the base grid layer for Profile Preferences ' in the current project.
    Dim prefPrj As AeccPreferencesProject
    Set prefPrj = AeccApplication.ActiveProjectPreferences

    MsgBox "The base grid layer for Profile Preferences in the current Project is: " & prefPrj.Profile.GetString(kBaseGridLayer), vbInformation, "Profile Example"

End Sub
Sub Example_ProfileBlocks()

' This example returns the count of alignment profiles in
' the active document.
Dim doc As AeccDocument
Set doc = AeccApplication.ActiveDocument

MsgBox "The number of alignment profile blocks in the current document is: "
    doc.ProfileBlocks.Count, vbInformation, "ProfileBlocks Example"

End Sub
ProfileByType Example

Examples:

Example_ProfileByType_EGProfiles()

' This example gets the surface name of the first existing ground profile
' in the first alignment in the collection. An existing ground profile is set based
' upon the surface name with a center type.
Dim align As AeccAlignment
Dim EGProf As AeccEGProfile
Dim cenEGProf As AeccEGProfile
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set EGProf = align.EGProfiles.Item(0)

Dim surfName As String
surfName = EGProf.SurfaceName

Set cenEGProf = align.EGProfiles.ProfileByType(kEgCenter, surfName)
MsgBox "The first station for the center existing ground profile is: " _
   & cenEGProf.StationElevations(0), vbInformation, "ProfileByType Example"

Example_ProfileByType_FGProfiles()

' This example uses the ProfileByType method to get the finished ground profil
' based on a center type, for the first alignment in the collection.
Dim align As AeccAlignment
Dim FGProf As AeccFGProfile
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set FGProf = align.FGProfiles.Item(0)

Set FGProf = align.FGProfiles.ProfileByType(kEgCenter)

MsgBox "The Type of the finished ground profile is: " _ & FGProf.Type, vbInformation, "ProfileByType Example"

End Sub
Sub Example_ProgramPath()

    ' This example returns the ProgramPath setting.
    Dim prefFiles As AeccPreferencesFiles
    Set prefFiles = AeccApplication.Preferences.Files

    MsgBox "The current value for ProgramPath is: " & prefFiles.ProgramPath, _
          vbInformation, "ProgramPath Example"

End Sub
Sub Example_ProjectName()

' This example displays the ProjectName setting for the current drawing
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.preferences

MsgBox "The setting for ProjectName is: " & dbPref.ProjectName, _
    vbInformation, "ProjectName Example"

End Sub
**ProjectPath Example**

**Examples:**

1. **PreferencesFiles**

2. **Projects**

---

**Sub** Example_ProjectPath_PreferencesFiles()

' This example returns the ProjectPath setting.
Dim prefFiles As AeccPreferencesFiles  
Set prefFiles = AeccApplication.Preferences.Files

MsgBox "The current value for ProjectPath is: " & prefFiles.ProjectPath, _  
vbInformation, "ProjectPath Example"

End Sub

---

**Sub** Example_ProjectPath_Projects()

' This example returns the ProjectPath setting.
Dim projs As AeccProjects  
Set projs = AeccApplication.Projects

MsgBox "The ProjectPath value for Projects is: " & projs.ProjectPath, _  
vbInformation, "ProjectPath Example"

End Sub
Sub Example_Projects()

    ' This example returns the number of Projects at the ProjectPath.
    Dim projs As AeccProjects
    Set projs = AeccApplication.Projects

    MsgBox "The number of projects at " & projs.ProjectPath & " is: " & projs.Count,
    vbInformation, "Projects Example"

End Sub
Sub Example_PrototypeName()

    ' This example returns the PrototypeName setting for the first Project 
    ' in the collection
    Dim proj As AeccProject
    Set proj = AeccApplication.Projects.Item(0)

    MsgBox "The PrototypeName for the first Project in the collection is: " & proj.
        , vbInformation, "PrototypeName Example"

End Sub
PrototypePath Example

Sub Example_PrototypePath()
    ' This example returns the PrototypePath setting.
    Dim prefFiles As AeccPreferencesFiles
    Set prefFiles = AeccApplication.Preferences.Files
    MsgBox "The current value for PrototypePath is: " & prefFiles.PrototypePath,
    vbInformation, "PrototypePath Example"
End Sub
Sub Example_Prototypes()

' This example returns the number of Prototypes at the PrototypePath
Dim prots As AeccPrototypes
Set prots = AeccApplication.Prototypes

MsgBox "The number of prototypes is " & prots.Count

End Sub
Sub Example_PVIs()

    ' This example uses the PVIs property to get the number of PVIs for the
    ' finished ground profile in the first alignment in the collection.
    Dim align As AeccAlignment
    Dim FGProf As AeccFGProfile
    Set align = AeccApplication.ActiveProject.Alignments.Item(0)
    Set FGProf = align.FGProfiles.Item(0)

    MsgBox "The number of PVIs for the finished ground profile in the first alignn
    & FGProf.PVIs.Count, vbInformation, "PVIs Example"

End Sub
Sub Example_RadialDistance()

' This example returns the RadialDistance for the first Spiral found in the first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Spiral entity in the first Alignment."

' Find first Spiral in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kSpiral Then
        alignMsg = "The RadialDistance for the first Spiral in the alignment is: " & alignEnt.RadialDistance
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "RadialDistance Example"

End Sub
Radius Example

Examples:

1 AlignCurve
1 ParcelCurve

Sub Example_Radius_AlignCurve()

' This example returns the Radius for the first Curve found in the
' first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Curve entity in the first Alignment."

' Find first Curve in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kCurve Then
        alignMsg = "The Radius for the first Curve in the alignment is: " _
        & Format(alignEnt.Radius, "0.00")
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "Radius Example"

End Sub

Sub Example_Radius_ParcelCurve()

' This example returns the Radius for the first curve in the
' first Parcel in the collection.


Dim parcel As AeccParcel
Dim parcelEnt As AeccParcelEntity
Set parcel = AeccApplication.ActiveProject.Parcels.Item(0)
Set parcelEnt = parcel.ParcelEntities.Item(0)

Dim parcelMsg As String
parcelMsg = "There is no Curve entity in the first Parcel."

' Find first Curve in the parcel
For Each parcelEnt In parcel.ParcelEntities
    If parcelEnt.Type = kParcelCurve Then
        parcelMsg = "The Radius for the first Curve in the Parcel is: " & Format(parcelEnt.Radius, "0.00")
        Exit For
    End If
End For

MsgBox parcelMsg, vbInformation, "Radius Example"

End Sub
Sub Example_RawDescription()

    ' This example returns the RawDescription for the first CogoPoint in the collec
    Dim cogoPnt As AeccCogoPoint
    Set cogoPnt = AeccApplication.ActiveProject.CogoPoints.Item(0)

    MsgBox "The RawDescription for the first CogoPoint in the collection is: " & .
    cogoPnt.RawDescription, vbInformation, "RawDescription Example"

End Sub
RemoveAll Example

Examples:

1 Alignment

1 StationEquations

Sub Example_RemoveAll_Alignment()

' This example adds an Alignment made up of a tangent.
Dim aligns As AeccAlignments
Dim align As AeccAlignment
Set aligns = AeccApplication.ActiveProject.Alignments

' Add an Alignment named "Example Alignment" and starting at Station 50.0
Set align = aligns.Add("Example Alignment", 50#)

' Add a tangent
Dim tangent As AeccAlignTangent
Set tangent = align.AddTangent(0#, 0#, 150#, 0#)

MsgBox "The total number of entities in the Alignment is: " & align.AlignEntities.Count

' Remove the entities from the alignment
align.RemoveAll

MsgBox "The total number of entities in the Alignment is: " & align.AlignEntities.Count

End Sub

Sub Example_RemoveAll_StationEquations()

' This example removes all StationEquations from
' the first alignment in the collection.
Dim aligns As AeccAlignments

'
Dim align As AeccAlignment
Set aligns = AeccApplication.ActiveProject.Alignments

' Get the first alignment
Set align = aligns.Item(0)

' Set the first alignment current.
aligns.CurrentAlignment = align.Name

' Show then StationEquation count for the first alignment
MsgBox "The StationEquation count for the first alignment is: " & align.StationEquations.Count,
vbInformation,"RemoveAll Example"

' Remove all StationEquations
align.StationEquations.RemoveAll

' Show the StationEquation count as zero after the remove all
MsgBox "The StationEquation count for the first alignment is: " & align.StationEquations.Count,
vbInformation,"RemoveAll Example"

End Sub
Sub Example_RemoveAllLabels()
  
    ' This example removes all labels from a selected contour

    On Error Resume Next

    ' Delete existing SelectionSet
    ThisDrawing.SelectionSets("SSet").Delete

    ' Create the selection set based on a point selection
    ' and filter for Contour objects
    Dim ssetObj As AcadSelectionSet
    Set ssetObj = ThisDrawing.SelectionSets.Add("SSet")

    Dim mode As Integer
    Dim gpCode(0) As Integer
    Dim dataValue(0) As Variant

    gpCode(0) = 0
    dataValue(0) = "AECC_CONTOUR"

    Dim groupCode As Variant
    Dim dataCode As Variant
    Dim returnPnt As Variant

    groupCode = gpCode
    dataCode = dataValue
    returnPnt = ThisDrawing.Utility.GetPoint(, "Select a contour line: ")
    ssetObj.SelectAtPoint returnPnt, groupCode, dataCode

    Dim objContour As AeccContour
    Set objContour = ssetObj.Item(0)

    objContour.RemoveAllLabels
End Sub
Sub Example_RemoveLabelAt()

    ' This example removes a label from a Contour object by selecting the contour

    On Error Resume Next

    ' Delete existing SelectionSet
    ThisDrawing.SelectionSets("SSet").Delete

    ' Create the selection set based on a point selection and filter for Contour objects
    Dim ssetObj As AcadSelectionSet
    Set ssetObj = ThisDrawing.SelectionSets.Add("SSet")

    Dim mode As Integer
    Dim gpCode(0) As Integer
    Dim dataValue(0) As Variant
    gpCode(0) = 0
    dataValue(0) = "AECC_CONTOUR"

    Dim groupCode As Variant
    Dim dataCode As Variant
    Dim returnPnt As Variant

    groupCode = gpCode
    dataCode = dataValue
    returnPnt = ThisDrawing.Utility.GetPoint(, "Enter a point on a contour line: ")
    ssetObj.SelectAtPoint returnPnt, groupCode, dataCode

    Dim Ent As AeccContour
    Set Ent = ssetObj.Item(0)
    Ent.RemoveLabelAt returnPnt(0), returnPnt(1)
End Sub
Renew Example

Examples:

1 Parcels

1 Surfaces

Sub Example_Rename_Parcels()

' This example starts by creating a Parcel named "NewParcel".
' The new Parcel name displayed. Finally, the new Parcel is renamed
' to "OldName" and is displayed again.
Dim parcels As AeccParcels
Dim parcel As AeccParcel
Set parcels = AeccApplication.ActiveProject.Parcels

' ' Add a new Parcel name "NewParcel"
Set parcel = parcels.Add("NewParcel")

MsgBox "The Parcel name is: " & parcel.Name, vbInformation, "Rename Exar

' ' Rename the new Parcel to "OldParcel"
parcels.Rename "NewParcel", "OldParcel"

MsgBox "The Parcel name is: " & parcel.Name, vbInformation, "Rename Exar

End Sub

Sub Example_Rename_Surfaces()

' This example renames a the first Surface in the collection to "FinalSurface".
Dim surfs As AeccSurfaces
Dim surf As AeccSurface
Set surfs = AeccApplication.ActiveProject.Surfaces

'
' Get the name of the first surface in the collection
Set surf = surfys.Item(0)
Dim surfName As String
surfName = surf.Name

' Rename the surface
surfs.Rename surfName, "FinalSurface"

MsgBox "The Surface name FinalSurface was named: " & surfName, vbInform

End Sub
Sub Example_RevisionNumber()

    ' This example returns the RevisionNumber for the first surface in the collection
    Dim surf As AeccSurface
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

    MsgBox "The RevisionNumber for the first surface is: " & surf.RevisionNumber

End Sub
Sub Example_RightWidth()

' This example returns the right width for the first alignment cross section in the collection.
Dim alignXsects As AeccCrossSectionBlocks
Dim alignXsect As AeccCrossSectionBlock
Set alignXsects = AeccApplication.ActiveDocument.CrossSectionBlocks
Set alignXsect = alignXsects.Item(0)

' Get the station for the first alignment cross section in the collection
Dim station As String
station = alignXsect.station

MsgBox "The right width for the alignment cross section at station " & station & Format(alignXsect.RightWidth, "0.00"), vbInformation, "RightWidth Example"

End Sub
Sub Example_RotateByDescriptionParam()

    ' This example returns the RotateByDescriptionParam for the first
    ' DescriptionKey in the first DescriptionKeyFile in the collection.
    Dim dKeyFile As AeccDescriptionKeyFile
    Dim dKey As AeccDescriptionKey
    Set dKeyFile = AeccApplication.ActiveProject.DescriptionKeyFiles.Item(0)
    Set dKey = dKeyFile.Item(0)

    MsgBox "The value for DescriptionKey RotateByDescriptionParam is: " & dK
    , vbInformation, "RotoateByDescriptionParam Example"

End Sub
Sub Example_RotateByFixedFactor()

' This example returns the RotateByFixedFactor for the first
' DescriptionKey in the first DescriptionKeyFile in the collection.
Dim dKeyFile As AeccDescriptionKeyFile
Dim dKey As AeccDescriptionKey
Set dKeyFile = AeccApplication.ActiveProject.DescriptionKeyFiles.Item(0)
Set dKey = dKeyFile.Item(0)

MsgBox "The value for DescriptionKey RotateByFixedFactor is: " & dKey.RotateByFixedFactor, vbInformation, "RotateByFixedFactor Example"

End Sub
Sub Example_RotateClockwise()

    ' This example returns the RotateClockwise for the first
    ' DescriptionKey in the first DescriptionKeyFile in the collection.
    Dim dKeyFile As AeccDescriptionKeyFile
    Dim dKey As AeccDescriptionKey
    Set dKeyFile = AeccApplication.ActiveProject.DescriptionKeyFiles.Item(0)
    Set dKey = dKeyFile.Item(0)

    MsgBox "The value for DescriptionKey RotateClockwise is: " & dKey.Rotate,
            , vbInformation, "RotateClockwise Example"

End Sub
Sub Example_RotateDescriptionParam()

' This example returns the RotateDescriptionParam for the first DescriptionKey in the first DescriptionKeyFile in the collection.
Dim dKeyFile As AeccDescriptionKeyFile
Dim dKey As AeccDescriptionKey
Set dKeyFile = AeccApplication.ActiveProject.DescriptionKeyFiles.Item(0)
Set dKey = dKeyFile.Item(0)

MsgBox "The value for DescriptionKey RotateDescriptionParam is: " & dKey.
    vbInformation, "RotateDescriptionParam Example"

End Sub
Sub Example_RotateFixedFactor()

' This example returns the RotateFixedFactor for the first DescriptionKey in the first DescriptionKeyFile in the collection.
Dim dKeyFile As AeccDescriptionKeyFile
Dim dKey As AeccDescriptionKey
Set dKeyFile = AeccApplication.ActiveProject.DescriptionKeyFiles.Item(0)
Set dKey = dKeyFile.Item(0)

MsgBox "The value for DescriptionKey RotateFixedFactor is: " & dKey.RotateFixedFactor, vbInformation, "RotateFixedFactor Example"

End Sub
Sub Example_SampleElevation()

' This example returns an array of points between a selected start and end point
' for the first surface in the collection. The array from points includes the start
' and end points and all points that intersect an edge between the start and
' endpoint.
Dim surf As AeccSurface
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

Dim pnt1 As Variant
Dim pnt2 As Variant
Dim derivedPnts As Variant

' Return points using a prompt
pnt1 = ThisDrawing.Utility.GetPoint(, "Select first point: ")
pnt2 = ThisDrawing.Utility.GetPoint(pnt1, "Select second point: ")

derivedPnts = surf.SampleElevations(pnt1, pnt2)

MsgBox "The first two points in the derived points are:" & vbCrLf & _
    derivedPnts(0) & ", " & derivedPnts(1) & ", " & derivedPnts(2) & vbCrLf & _
    derivedPnts(3) & ", " & derivedPnts(4) & ", " & derivedPnts(5), vbInformation & 
"SampleElevations Example"

End Sub
Sub Example_Save_Alignment()

' This example changes the Description and StartingStation
' for the first alignment in the collection.
Dim aligns As AeccAlignments
Dim align As AeccAlignment
Set aligns = AeccApplication.ActiveProject.Alignments
Set align = aligns.Item(0)

' Make the first alignment in the collection current
aligns.CurrentAlignment = align.Name

' Set AutoSave to FALSE to prevent writing the changes
' until they are all made
aligns.AutoSave = False

' Make changes
align.Description = "New Description"
align.StartingStation = 20#

' Save changes
align.Save

' Reset AutoSave to the default state of TRUE
aligns.AutoSave = True
Sub Example_Save_CogoPoint()

' This example changes the Northing, Easting, and Elevation
' for the first CogoPoint in the collection.
Dim cogoPnts As AeccCogoPoints
Dim cogoPnt As AeccCogoPoint
Set cogoPnts = AeccApplication.ActiveProject.CogoPoints
Set cogoPnt = cogoPnts.Item(0)

' Set AutoSave to FALSE to prevent writing the changes
' until they are all made
cogoPnts.AutoSave = False

' Make changes
cogoPnt.Northing = 100#
cogoPnt.Easting = 100#
cogoPnt.Elevation = 150#

' Save changes
cogoPnt.Save

' Reset AutoSave to the default state of TRUE
cogoPnts.AutoSave = True

MsgBox "The following changes were made to CogoPoint " & cogoPnt.Number:
" Northing: " & cogoPnt.Northing & vbCrLf & _
" Easting: " & cogoPnt.Easting & vbCrLf & _
" Elevation: " & cogoPnt.Elevation, vbInformation, "Save Example"
End Sub
Sub Example_Save_DescriptionKey()

' This example changes the layer settings
' for the first DescriptionKey in the collection.
Dim dKeyFiles As AeccDescriptionKeyFiles
Dim dKeyFile As AeccDescriptionKeyFile
Dim dKey As AeccDescriptionKey
Set dKeyFiles = AeccApplication.ActiveProject.DescriptionKeyFiles
Set dKeyFile = dKeyFiles.Item(0)
Set dKey = dKeyFile.Item(0)

' Set AutoSave to FALSE to prevent writing the changes
' until they are all made
dKeyFiles.AutoSave = False

' Make changes
dKey.DescriptionLayer = "dKey Desc"
dKey.SymbolLayer = "dKey Symb"

' Save changes
dKey.Save

' Reset AutoSave to the default state of TRUE
dKeyFiles.AutoSave = True

MsgBox "The DescriptionLayer is: " & dKey.DescriptionLayer & vbCrLf & 
"The SymbolLayer is: " & dKey.SymbolLayer, vbInformation, "Save Examp

End Sub

Sub Example_Save_PointGroup()

' This example changes the Description and Elevation
' for the first PointGroup in the collection.
Dim pntGrps As AeccPointGroups
Dim pntGrp As AeccPointGroup
Set pntGrps = AeccApplication.ActiveProject.PointGroups
Set pntGrp = pntGrps.Item(0)

' Set AutoSave to FALSE to prevent writing the changes
' until they are all made
pntGrps.AutoSave = False

' Make changes
pntGrp.Description = "New Description"
pntGrp.Elevation = 100#

' Save changes
pntGrp.Save

' Reset AutoSave to the default state of TRUE
pntGrps.AutoSave = True

MsgBox "The first PointGroup Description is: " & pntGrp.Description & vbCrLf
     "The first PointGroup Elevation is: " & pntGrp.Elevation, vbInformation, "S

End Sub
SaveAsDefault Example

Sub Example_SaveAsDefault()

' This example sets the TextHeight setting to 0.25
' and saves it as the default.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.preferences

dbPref.TextHeight = 0.25
dbPref.SaveAsDefault

MsgBox "The new default value for TextHeight is: " & dbPref.TextHeight, _
    vbInformation, "SaveAsDefault Example"

End Sub
SaveSetupProfile Example

Sub Example_SaveSetupProfile()

' This example saves a new drawing setup profile.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.DatabasePreferences

' The file extension ".set" is appended automatically.
dbPref.SaveSetupProfile("myprofile")

End Sub
Sub Example_SectionVolume()

' This example calculates the volume data for the second cross section in the
' cross section collection for the first alignment in the collection.
Dim aligns As AeccAlignments
Dim align As AeccAlignment
Dim xSect As AeccCrossSection
Set aligns = AeccApplication.ActiveProject.Alignments
Set align = aligns.Item(0)
Set xSect = align.CrossSections.Item(0)

' Get the cross section station and format it
Dim station As String
station = aligns.DoubleToStaFormat(xSect.station)

' Define result variables
Dim cutArea As Double
Dim cutCentroid As Double
Dim cutVolume As Double
Dim fillArea As Double
Dim fillCentroid As Double
Dim fillVolume As Double

' Get volume data
xSect.SectionVolume kVolumeByAvgEndArea, 1#, 1#, 1#, cutArea, cutCentroid

MsgBox "The volume data at station " & station & " is: " & vbCrLf & vbCrLf
  "Cut Area: " & Format(cutArea, "0.00") & vbCrLf & vbCrLf
  "Cut Centroid: " & Format(cutCentroid, "0.00") & vbCrLf & vbCrLf
  "Cut Volume: " & Format(cutVolume, "0.00") & vbCrLf & vbCrLf
  "Fill Area: " & Format(fillArea, "0.00") & vbCrLf & vbCrLf
  "Fill Centroid: " & Format(fillCentroid, "0.00") & vbCrLf & vbCrLf
  "Fill Volume: " & Format(fillVolume, "0.00"), vbInformation, "SectionVolume Example"
End Sub
Sub Example_ScaleByDescriptionParam()

' This example returns the ScaleByDescriptionParam for the first DescriptionKey in the first DescriptionKeyFile in the collection.
Dim dKeyFile As AeccDescriptionKeyFile
Dim dKey As AeccDescriptionKey
Set dKeyFile = AeccApplication.ActiveProject.DescriptionKeyFiles.Item(0)
Set dKey = dKeyFile.Item(0)

MsgBox "The value for DescriptionKey ScaleByDescriptionParam is: " & dKey.ScaleByDescriptionParam, vbInformation, "ScaleByDescriptionKey Example"

End Sub
Sub Example_ScaleByDrawingScale()

    ' This example returns the ScaleByDrawingScale for the first
    ' DescriptionKey in the first DescriptionKeyFile in the collection.
    Dim dKeyFile As AeccDescriptionKeyFile
    Dim dKey As AeccDescriptionKey
    Set dKeyFile = AeccApplication.ActiveProject.DescriptionKeyFiles.Item(0)
    Set dKey = dKeyFile.Item(0)

    MsgBox "The value for DescriptionKey ScaleByDrawingScale is: " & dKey.ScaleByDrawingScale, vbInformation, "ScaleByDrawingScale Example"

End Sub
Sub Example_ScaleByFixedFactor()

' This example returns the ScaleByFixedFactor for the first
' DescriptionKey in the first DescriptionKeyFile in the collection.
Dim dKeyFile As AeccDescriptionKeyFile
Dim dKey As AeccDescriptionKey
Set dKeyFile = AeccApplication.ActiveProject.DescriptionKeyFiles.Item(0)
Set dKey = dKeyFile.Item(0)

MsgBox "The value for DescriptionKey ScaleByFixedFactor is: " & dKey.ScaleByFixedFactor_, vbInformation, "ScaleByFixedFactor Example"

End Sub
Sub Example_ScaleDescriptionParam()

    ' This example returns the ScaleDescriptionParam for the first
    ' DescriptionKey in the first DescriptionKeyFile in the collection.
    Dim dKeyFile As AeccDescriptionKeyFile
    Dim dKey As AeccDescriptionKey
    Set dKeyFile = AeccApplication.ActiveProject.DescriptionKeyFiles.Item(0)
    Set dKey = dKeyFile.Item(0)

    MsgBox "The value for DescriptionKey ScaleDescriptionParam is: " & dKey.ScaleDescriptionParam,
        vbInformation, "ScaleDescriptionParam Example"

End Sub
Sub Example_ScaleFixedFactor()

' This example returns the ScaleFixedFactor for the first
' DescriptionKey in the first DescriptionKeyFile in the collection.
Dim dKeyFile As AeccDescriptionKeyFile
Dim dKey As AeccDescriptionKey
Set dKeyFile = AeccApplication.ActiveProject.DescriptionKeyFiles.Item(0)
Set dKey = dKeyFile.Item(0)

MsgBox "The value for DescriptionKey ScaleFixedFactor is: " & dKey.ScaleFixedFactor, vbInformation, "ScaleFixedFactor Example"

End Sub
Sub Example_ScaleInXY()

    ' This example returns the ScaleInXY for the first
    ' DescriptionKey in the first DescriptionKeyFile in the collection.
    Dim dKeyFile As AeccDescriptionKeyFile
    Dim dKey As AeccDescriptionKey
    Set dKeyFile = AeccApplication.ActiveProject.DescriptionKeyFiles.Item(0)
    Set dKey = dKeyFile.Item(0)

    MsgBox "The value for DescriptionKey ScaleInXY is: " & dKey.ScaleInXY, _
        vbInformation, "ScaleInXY Example"

End Sub
Sub Example_ScaleInZ()

  ' This example returns the ScaleInZ for the first
  ' DescriptionKey in the first DescriptionKeyFile in the collection.
  Dim dKeyFile As AeccDescriptionKeyFile
  Dim dKey As AeccDescriptionKey
  Set dKeyFile = AeccApplication.ActiveProject.DescriptionKeyFiles.Item(0)
  Set dKey = dKeyFile.Item(0)

  MsgBox "The value for DescriptionKey ScaleInZ is: " & dKey.ScaleInZ,
  _
  vbInformation, "ScaleInZ Example"

End Sub
Sub Example_ScaleOnInsert()

' This example returns the ScaleOnInsert setting for the current drawing.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.preferences

' Convert the scale on insert value to a string.
Dim scaleins As String
If dbPref.ScaleOnInsert = True Then
    scaleins = "scale on insert."
Else
    scaleins = "do not scale on insert."
End If

MsgBox "The current value for ScaleOnInsert is " & scaleins, vbInformation, "ScaleOnInsert Example"

End Sub
Sub Example_SearchType()

' This example returns how the Faces collection was generated.
Dim surf As AeccSurface
Dim faces As AeccFaces
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set faces = Surf.Outputs.Faces

Dim msgStr As String

Select Case faces.SearchType
Case kNoSearch
    msgStr = "No search"
Case kSearchByBoundar
    msgStr = "Search by boundary"
Case kSearchByPath
    msgStr = "Search by path"
Case kSearchByPoint
    msgStr = "Search by point"
End Select

MsgBox "The Faces search type is set to: " & msgStr, vbInformation, "SearchTy

End Sub
Sub Example_SectionByStation()

' This example returns a cross section by providing the station
' to the first alignment in the collection.
Dim align As AeccAlignment
Dim xSects As AeccCrossSections
Dim xSect As AeccCrossSection
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set xSects = align.CrossSections

' Get the station for the first cross section in the cross section collection
Dim station As Double
station = xSects.Item(0).station

' Get the cross section block for the station set above
Set xSect = xSects.SectionByStation(station)

MsgBox "The maximum elevation for the cross section at station " & station & ' 
    Format(xSect.MaxElevation, "0.00"), vbInformation, "SectionByStation Example"

End Sub
Sub Example_SetBoundingBox()

' This example sets a surface bounding box for the first surface
' in the collection, based upon two picked points.
Dim surf As AeccSurface
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

Dim pnt1 As Variant
Dim pnt2 As Variant

pnt1 = ThisDrawing.Utility.GetPoint(, "Select first corner : ")
pnt2 = ThisDrawing.Utility.GetCorner(Pnt1, "Select other corner: ")

surf.SetBoundingBox pnt1, pnt2

End Sub
SetDouble Example

Sub Example_SetDouble()

    ' This example uses SetDouble to define the ContLabelSpacingDist
    ' for PreferencesSurface.
    Dim surfPref As AeccPreferencesSurface
    Set surfPref = AeccApplication.ActiveProject.preferences.Surface

    surfPref.SetDouble kContLabelSpacingDist, 200#

    MsgBox "The value for ContLabelSpacingDist is " & surfPref.GetDouble(kCo
        vbInformation, "SetDouble Example"

End Sub
Sub Example_SetInteger()

    ' This example uses SetInteger to define the ContLabelPrecision
    ' for PreferencesSurface.
    Dim surfPref As AeccPreferencesSurface
    Set surfPref = AeccApplication.ActiveProject.preferences.Surface

    surfPref.SetInteger kContLabelPrecision, 3

    MsgBox "The value for ContLabelPrecision is " & surfPref.GetInteger(kContL
    vbInformation, "SetInteger Example"

End Sub
SetReferenceCurve Example

Sub Example_SetReferenceCurve()

' This example uses the SetReferenceCurve method to attach curve text
' to a selected arc or circle

Dim ent As AcadEntity
Dim basePnt As Variant

' Pick the curve
On Error Resume Next
RETRY:
Err.Clear
ThisDrawing.Utility.GetEntity ent, basePnt, "Select a curve"
If Err <> 0 Then
    ThisDrawing.Utility.Prompt "No entity found" + vbCrLf
    GoTo RETRY
End If
If ent.ObjectName <> "AcDbCircle" And _
    ent.ObjectName <> "AcDbArc" Then
    ThisDrawing.Utility.Prompt "Entity is not an arc or circle" + vbCrLf
    GoTo RETRY
End If

' Attach the curve text
Dim ctext As AeccCurveText
Set ctext = ThisDrawing.ModelSpace.AddCustomObject("AecDbCurveText")

ctext.SetReferenceCurve ent
ctext.TextAbove = "ABOVE"
ctext.TextBelow = "BELOW"

End Sub
Sub Example_SetString()

    ' This example uses SetString to define the SurfaceLayer
    ' for PreferencesSurface.
    Dim surfPref As AeccPreferencesSurface
    Set surfPref = AeccApplication.ActiveProject.preferences.Surface

    surfPref.SetString kSurfaceLayer, "MyLayer"

    MsgBox "The value for SurfaceLayer is " & surfPref.GetString(kSurfaceLayer)

End Sub
Sub Example_SheetHeight()

' This example returns the SheetHeight setting for the current drawing.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.Preferences

MsgBox "The current value for SheetHeight is: " & dbPref.SheetHeight, _
vbInformation, "SheetHeight Example"

End Sub
```
Sub Example_SheetWidth()

' This example returns the SheetWidth setting for the current drawing.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.Preferences

MsgBox "The current value for SheetWidth is " & dbPref.SheetWidth, _
     vbInformation, "SheetWidth Example"

End Sub
```
Sub Example_ShortTangent()

' This example returns the ShortTangent for the first Spiral found in the
' first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Spiral entity in the first Alignment."

' Find first Spiral in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kSpiral Then
        alignMsg = "The ShortTangent for the first Spiral in the alignment is: " &
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "ShortTangent Example"

End Sub
Sub Example_ShowStartupDialog()

    ' This example returns the ShowStartupDialog setting.
    Dim prefUser As AeccPreferencesUser
    Set prefUser = AeccApplication.Preferences.User

    MsgBox "The current value for ShowStartupDialog is: " & prefUser.ShowStart

End Sub
Sub Example_ShowSubfolders()

    ' This example returns the ShowSubfolders setting for the active project.
    Dim dwgs As AeccDrawings
    Set dwgs = AeccApplication.ActiveProject.Drawings

    MsgBox "The ShowSubfolders value for Drawings in the current Project is " & dwgs.ShowSubfolders,
            , vbInformation, "ShowSubfolders Example"

End Sub
Sub Example_SpeedTablesPath()

    ' This example returns the SpeedTablesPath setting.
    Dim prefFiles As AeccPreferencesFiles
    Set prefFiles = AeccApplication.Preferences.Files

    MsgBox "The current value for SpeedTablesPath is: " & prefFiles.SpeedTablesPath
    _
    vbInformation, "SpeedTablesPath Example"

End Sub
Sub Example_SpiEasting()

' This example returns the SpiEasting for the first Spiral found in the
' first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Spiral entity in the first Alignment."

' Find first Spiral in the alignment
For Each alignEnt In align.AlignEntities
  If alignEnt.Type = kSpiral Then
    alignMsg = "The SpiEasting for the first Spiral in the alignment is: " & ali
    Exit For
  End If
Next

MsgBox alignMsg, vbInformation, "SpiEasting Example"

End Sub
Sub Example_SpilTangent()

' This example returns the SpilTangent for the first Spiral found in the
' first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Spiral entity in the first Alignment."

' Find first Spiral in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kSpiral Then
        alignMsg = "The SpilTangent for the first Spiral in the alignment is: " & alignEnt.SpilTangent
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "SpilTangent Example"

End Sub
Sub Example_SpiNorthing()

' This example returns the SpiNorthing for the first Spiral found in the ' first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Spiral entity in the first Alignment."

' Find first Spiral in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kSpiral Then
        alignMsg = "The SpiNorthing for the first Spiral in the alignment is: " & alignEnt.SpiNorthing
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "SpiNorthing Example"

End Sub
Sub Example_SpiralType1()

' This example returns the SpiralType1 for the first Spiral found in the first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Spiral entity in the first Alignment."

' Find first Spiral in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kSpiral Then
        alignMsg = "The SpiralType1 for the first Spiral in the alignment is: " & a
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "SpiralType1 Example"

End Sub
Sub Example_SpiralType2()

    ' This example returns the SpiralType2 for the first Spiral found in the
    ' first Alignment in the collection.
    Dim align As AeccAlignment
    Dim alignEnts As AeccAlignEntity
    Set align = AeccApplication.ActiveProject.Alignments.Item(0)

    Dim alignMsg As String
    alignMsg = "There is no Spiral entity in the first Alignment."

    ' Find first Spiral in the alignment
    For Each alignEnt In align.AlignEntities
        If alignEnt.Type = kSpiral Then
            alignMsg = "The SpiralType2 for the first Spiral in the alignment is: " & a
            Exit For
        End If
    Next

    MsgBox alignMsg, vbInformation, "SpiralType2 Example"

End Sub
Sub Example_StartDirection_AlignCurve()

    ' This example returns the StartDirection for the first Curve found in the
    ' first Alignment in the collection.
    Dim align As AeccAlignment
    Dim alignEnts As AeccAlignEntity
    Set align = AeccApplication.ActiveProject.Alignments.Item(0)

    Dim alignMsg As String
    alignMsg = "There is no Curve entity in the first Alignment."

    ' Find first Curve in the alignment
    For Each alignEnt In align.AlignEntities
        If alignEnt.Type = kCurve Then
            alignMsg = "The StartDirection for the first Curve in the alignment is: " &
        Exit For
    End If
    Next

    MsgBox alignMsg, vbInformation, "StartDirection Example"
End Sub

Sub Example_StartDirection_AlignSpiral()

    ' This example returns the StartDirection for the first Spiral found in the
    ' first Alignment in the collection.
    Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Spiral entity in the first Alignment."

' Find first Spiral in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kSpiral Then
        alignMsg = "The StartDirection for the first Spiral in the alignment is: " &
    Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "StartDirection Example"

End Sub
StartEasting Example

Examples:

- AlignEntity, AlignCurve, AlignSpiral, AlignTangent
- ParcelEntity, ParcelCurve, ParcelLine

```vba
Sub Example_StartEasting_AlignEntity()

' This example returns the StartEasting for the first entity in the first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnt As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set alignEnt = align.AlignEntities.Item(0)

MsgBox "The StartEasting of the first entity in the Alignment is: " & alignEnt.StartEasting, vbInformation, "StartEasting Example"

End Sub
```

```vba
Sub Example_StartEasting_ParcelEntity()

' This example returns the StartEasting for the first entity in the first Parcel in the collection.
Dim parcel As AeccParcel
Dim parcelEnt As AeccParcelEntity
Set parcel = AeccApplication.ActiveProject.Parcels.Item(0)
Set parcelEnt = parcel.ParcelEntities.Item(0)

MsgBox "The StartEasting of the first entity in the Parcel is: " & parcelEnt.StartEasting, vbInformation, "StartEasting Example"

End Sub
```
StartingStation Example

Examples:

- AlignEntity, AlignCurve, AlignSpiral, AlignTangent
- Alignment
- ProfileBlock (Civil Engineering Feature)

---

Sub Example_StartingStation_AlignEntity()

' This example returns the StartingStation for the first entity in the first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnt As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set alignEnt = align.AlignEntities.Item(0)

MsgBox "The StartingStation of the first entity in the Alignment is: " & _
    alignEnt.StartingStation, vbInformation, "StartingStation Example"

End Sub

---

Sub Example_StartingStation_Alignment()

' This example returns the StartingStation for the first alignment in the collection.
Dim align As AeccAlignment
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

MsgBox "The StartingStation of the first Alignment is: " & align.StartingStation
    vbInformation, "StartingStation Example"

End Sub
Sub Example_StartingStation_ProfileBlock()

' This example returns the StartingStation for the first ProfileBlock
' in the collection
Dim alignProf As AeccProfileBlock
Set alignProf = AeccApplication.ActiveDocument.ProfileBlocks.Item(0)

MsgBox "The StartingStation for the first ProfileBlock in the collection is: " _
     & alignProf.StartingStation, vbInformation, "StartingStation Example"

End Sub
StartNorthing Example

Examples:

- AlignEntity, AlignCurve, AlignSpiral, AlignTangent
- ParcelEntity, ParcelCurve, ParcelLine

Sub Example_StartNorthing_AlignEntity()

' This example returns the StartNorthing for the first entity in the first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnt As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set alignEnt = align.AlignEntities.Item(0)

MsgBox "The StartNorthing of the first entity in the Alignment is: " & alignEnt.StartNorthing, vbInformation, "StartNorthing Example"

End Sub

Sub Example_StartNorthing_ParcelEntity()

' This example returns the StartNorthing for the first entity in the first Parcel in the collection.
Dim parcel As AeccParcel
Dim parcelEnt As AeccParcelEntity
Set parcel = AeccApplication.ActiveProject.Parcels.Item(0)
Set parcelEnt = parcel.ParcelEntities.Item(0)

MsgBox "The StartNorthing of the first entity in the Parcel is: " & parcelEnt.StartNorthing, vbInformation, "StartNorthing Example"

End Sub
Station Example

Examples:

1. [CrossSection](Civil Engineering Feature)
1. [CrossSectionBlock](Civil Engineering Feature)
1. [PVI](Civil Engineering Feature)
1. [Superelevation](Civil Engineering Feature)

Sub Example_Station_CrossSection()

' This example returns the station for the first cross section in the
' cross section collection for the first alignment in the collection.
Dim aligns As AeccAlignments
Dim align As AeccAlignment
Dim xSect As AeccCrossSection
Set aligns = AeccApplication.ActiveProject.Alignments
Set align = aligns.Item(0)
Set xSect = align.CrossSections.Item(0)

' Get the cross section station and format it
Dim station As String
station = aligns.DoubleToStaFormat(xSect.station)

MsgBox "The station for the first cross section in the collection is: " & _
    station, vbInformation, "Station Example"

End Sub

Sub Example_Station_CrossSectionBlock()

' This example returns the station for the first alignment cross section in the collection.
Dim aligns As AeccAlignments
Dim alignXsects As AeccCrossSectionBlocks
Dim alignXSect As AeccCrossSectionBlock
Set aligns = AeccApplication.ActiveProject.Alignments
Set alignXSects = AeccApplication.ActiveDocument.CrossSectionBlocks
Set alignXSect = alignXSects.Item(0)

' Get the station for the first alignment cross section in the collection
Dim station As String
station = aligns.DoubleToStaFormat(alignXSect.station)

MsgBox "The station for the first alignment cross section in the collection is: " & station, vbInformation, "Station Example"

End Sub

Sub Example_Station_PVI()

' This example returns the Station value for the second PVI in the first finished ground of the first alignment in the collection.
Dim align As AeccAlignment
Dim FGProf As AeccFGProfile
Dim PVI As AeccPVI
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set FGProf = align.FGProfiles.Item(0)
Set PVI = FGProf.PVIs.Item(1)

MsgBox "The Station of the second PVI is: " & Format(PVI.Station, "0.000"), vbInformation, "Station Example"

End Sub

Sub Example_Station_Superelevation()

' This example returns the data for the first superelevation in the superelevation collection for the first alignment in the collection.
Dim aligns As AeccAlignments
Dim align As AeccAlignment
Dim sElev As AeccSuperelevation
Set aligns = AeccApplication.ActiveProject.Alignments
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set sElev = align.Superelevations.Item(0)

' Get the alignment name
Dim alignName As String
alignName = align.Name

' Get the superelevation station and format it
Dim station As String
station = aligns.DoubleToStaFormat(sElev.station)

' Get the superelevation curve code
Dim crvCode As String
Select Case sElev.curveCode
    Case kSERightHandCurve
        crvCode = "Right Hand Curve"
    Case kSELeftHandCurve
        crvCode = "Left Hand Curve"
End Select

' Get the superelevation code
Dim supCode As String
Select Case sElev.SuperelevationCode
    Case kSEFullCrown
        supCode = "Full Crown"
    Case kSEHalfCrown
        supCode = "Half Crown"
    Case kSECrownRemoved
        supCode = "Crown Removed"
    Case kSEFullSuperelevations
        supCode = "Full Superelevation"
    Case kSEReverseCurve
        supCode = "Reverse Curve"
    Case kSECompoundCurve
        supCode = "Compound Curve"
End Select

MsgBox "The alignment name is: " & alignName & vbCrLf & _
"The data for the first superelevation is: " & vbCrLf & _
vbTab & "Station: " & station & vbCrLf & _
vbTab & "Curve Code: " & crvCode & vbCrLf & _
vbTab & "Superelevation Code: " & supCode, _
vbInformation, "Station Example"

End Sub
Sub Example_StationAhead()

    ' This example returns the StationAhead value
    ' for the StationEquation in the collection.
    Dim align As AeccAlignment
    Dim StaEqu As AeccStationEquation
    Set align = AeccApplication.ActiveProject.Alignments.Item(0)
    Set StaEqu = align.StationEquations.Item(0)

    MsgBox "The setting StationAhead is " & StaEqu.StationAhead, vbInformation

End Sub
Sub Example_StationBack()

    ' This example returns the StationBack value
    ' for the StationEquation in the collection.
    Dim align As AeccAlignment
    Dim StaEqu As AeccStationEquation
    Set align = AeccApplication.ActiveProject.Alignments.Item(0)
    Set StaEqu = align.StationEquations.Item(0)

    MsgBox "The setting StationBack is " & StaEqu.StationBack, vbInformation, '

End Sub
StationElevations Example

Sub Example_StationElevations()

' This example uses StationElevations to displays the station and elevation
' for the first existing ground profile in the first alignment in the collection.
Dim align As AeccAlignment
Dim EGProf As AeccEGProfile
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set EGProf = align.EGProfiles.Item(0)

MsgBox "The first station for the first existing ground profile is: " _
& Format(EGProf.StationElevations(0), "0.00") & vbCrLf & _
"The first elevation for the first existing ground profile is: " _
& Format(EGProf.StationElevations(1), "0.00") & vbCrLf, _
vbInformation, "StaionElevations Example"

End Sub
Sub Example_StationElevationToXy()

' This example returns the StationElevationToXy for the first ProfileBlock
' in the collection.
Dim alignProf As AeccProfileBlock
Set alignProf = AeccApplication.ActiveDocument.ProfileBlocks.Item(0)

Dim StaElev(0 To 1) As Double
Dim XY As Variant
Dim station As Double
Dim elevation As Double

'Get the station and the elevation
station = ThisDrawing.Utility.GetReal("Enter the station for the X value: ")
elevation = ThisDrawing.Utility.GetReal("Enter the elevation for the Y value: ")

' Use the entered values to get the X and Y values
staElev(0) = station
staElev(1) = elevation

XY = alignProf.StationElevationToXy(staElev)

MsgBox "The X value for the StationElevation is: " & Format(XY(0), "0.00")
"The Y value for the StationElevation is: " & Format(XY(1), "0.00")
, vbInformation, "StationElevationToXy Example"

End Sub
Sub Example_StationEquations()

    ' This example returns the number of StationEquations for the
    ' first Alignment in the collection.
    Dim align As AeccAlignment
    Dim staEqu As AeccStationEquations
    Set align = AeccApplication.ActiveProject.Alignments.Item(0)
    Set staEqu = align.StationEquations

    MsgBox "The number of StationEquations in this project is: " & staEqu.Count

End Sub
Sub Example_StationIncrement()

' This example returns the StationIncrement for the first ProfileBlock in the collection
Dim alignProf As AeccProfileBlock
Set alignProf = AeccApplication.ActiveDocument.ProfileBlocks.Item(0)

MsgBox "The StationIncrement for the first ProfileBlock in the collection is: " & alignProf.StationIncrement, vbInformation, "StationIncrement Example"

End Sub
Sub Example_StationOffset()

' This example gets the StationOffset from user supplied point
' for the first Alignment in the collection..
Dim align As AeccAlignment
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim returnPnt As Variant
Dim station As Double
Dim offset As Double
Dim direction As Double

' Get the point
returnPnt = ThisDrawing.Utility.GetPoint(, "Enter a point near the first alignmen"

' Get the station offset location
align.StationOffset returnPnt(0), returnPnt(1), station, offset, direction

MsgBox "The StationOffset for the first Alignment is:" & vbCrLf & _
"  Station: " & station & vbCrLf & _
"  Offset: " & offset & vbCrLf & _
"  Direction: " & direction, vbInformation, "StationOffset Example"

End Sub
Sub Example_Status()

' This example displays the Status for the first surface in the collection.
Dim surf As AeccSurface
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

Dim strStatus As String

' Set string to status of surface
Select Case surf.Status
Case kNoData
    strStatus = "has no data."
Case kNotBuilt
    strStatus = "is not built."
Case kUpToDate
    strStatus = "is up to date."
Case kOutOfDate
    strStatus = "is out of date."
End Select

MsgBox "The current surface " & strStatus, vbInformation, "Status Example"

End Sub
' This example returns the data for the first superelevation in the ' superelevation collection for the first alignment in the collection.
Dim aligns As AeccAlignments
Dim align As AeccAlignment
Dim sElev As AeccSuperelevation
Set aligns = AeccApplication.ActiveProject.Alignments
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set sElev = align.Superelevations.Item(0)

' Get the alignment name
Dim alignName As String
alignName = align.Name

' Get the superelevation station and format it
Dim station As String
station = aligns.DoubleToStaFormat(sElev.station)

' Get the superelevation curve code
Dim crvCode As String
Select Case sElev.curveCode
Case kSERightHandCurve
   crvCode = "Right Hand Curve"
Case kSELeftHandCurve
   crvCode = "Left Hand Curve"
End Select

' Get the superelevation code
Dim supCode As String
Select Case sElev.SuperelevationCode
Case kSEFullCrown
   supCode = "Full Crown"
Case kSEHalfCrown
supCode = "Half Crown"
Case kSECCrownRemoved
    supCode = "Crown Removed"
Case kSECCFullSuperelevations
    supCode = "Full Superelevation"
Case kSERCCReverseCurve
    supCode = "Reverse Curve"
Case kSECCompoundCurve
    supCode = "Compound Curve"
End Select

MsgBox "The alignment name is: " & alignName & vbCrLf & _
    "The data for the first superelevation is: " & vbCrLf & _
    vbTab & "Station: " & station & vbCrLf & _
    vbTab & "Curve Code: " & crvCode & vbCrLf & _
    vbTab & "Superelevation Code: " & supCode, _
    vbInformation, "SuperelevationCode Example"

End Sub
Superelevations Example

Sub Example_Superelevations()

    ' This example returns the number of superelevations for the
    ' first alignment in the collection.
    Dim align As AeccAlignment
    Set align = AeccApplication.ActiveProject.Alignments.Item(0)

    MsgBox "The number of superelevations in first alignment is: " _
        & align.Superelevations.Count, vbInformation, "Superelevations Example"

End Sub
Sub Example_Surface()

    ' This example returns the PreferencesSurface object that
    ' is used to access the SurfaceLayer property
    Dim surfPref As AeccPreferencesSurface
    Set surfPref = AeccApplication.ActiveProject.preferences.Surface

    MsgBox "The Surface preferences for SurfaceLayer is: " & surfPref.GetString(vbInformation, "Surface Example"

End Sub
Sub Example_SurfaceName()

    ' This example uses SurfaceName to displays the name of the surface
    ' for the first existing ground profile in the first alignment in the collection.
    Dim align As AeccAlignment
    Dim EGProf As AeccEGProfile
    Set align = AeccApplication.ActiveProject.Alignments.Item(0)
    Set EGProf = align.EGProfiles.Item(0)

    MsgBox "The SurfaceName for the first existing ground profile is: " _
    & EGProf.SurfaceName, vbInformation, "SurfaceName Example"

End Sub
Sub Example_Surfaces()

' This example returns the vertical exaggeration factor for Surfaces Preferences in the current project.
Dim prefPrj As AeccPreferencesProject
Set prefPrj = AeccApplication.ActiveProject.Preferences

MsgBox "The vertical exaggeration factor for Surfaces Preferences in the current project is: " & prefPrj.Surface.GetDouble(kVertExaggerationFactor), vbInformation, "Surfaces Example"

End Sub
SymbolBlock Example

Sub Example_SymbolBlock()

' This example returns the SymbolBlock for the first
' DescriptionKey in the first DescriptionKeyFile in the collection.
Dim dKeyFile As AeccDescriptionKeyFile
Dim dKey As AeccDescriptionKey
Set dKeyFile = AeccApplication.ActiveProject.DescriptionKeyFiles.Item(0)
Set dKey = dKeyFile.Item(0)

MsgBox "The value for DescriptionKey SymbolBlock is " & dKey.SymbolBlock,
vbInformation, "SymbolBlock Example"

End Sub
Sub Example_SymbolLayer()

' This example returns the SymbolLayer for the first DescriptionKey in the first DescriptionKeyFile in the collection.
Dim dKeyFile As AeccDescriptionKeyFile
Dim dKey As AeccDescriptionKey
Set dKeyFile = AeccApplication.ActiveProject.DescriptionKeyFiles.Item(0)
Set dKey = dKeyFile.Item(0)

MsgBox "The value for DescriptionKey SymbolLayer is " & dKey.SymbolLayer,
        vbInformation, "SymbolLayer Example"

End Sub
SymbolManagerPath Example

Sub Example_SymbolManagerPath()

' This example returns the SymbolManagerPath setting.
Dim prefFiles As AeccPreferencesFiles
Set prefFiles = AeccApplication.Preferences.Files

MsgBox "The current value for SymbolManagerPath is: " & prefFiles.SymbolManagerPath, vbInformation, "SymbolManagerPath Example"

End Sub
Sub Example_SystemPath()

    ' This example returns the SystemPath setting.
    Dim prefFiles As AeccPreferencesFiles
    Set prefFiles = AeccApplication.Preferences.Files

    MsgBox "The current value for SystemPath is: " & prefFiles.SystemPath, _
            vbInformation, "SystemPath Example"

End Sub
Sub Example_TangentLabelIncrement()

' This example returns the TangentLabelIncrement for the first ProfileBlock in the collection
Dim alignProf As AeccProfileBlock
Set alignProf = AeccApplication.ActiveDocument.ProfileBlocks.Item(0)

MsgBox "The TangentLabelIncrement for the first ProfileBlock in the collection is:
& alignProf.TangentLabelIncrement, vbInformation, "TangentLabelIncrement Example"

End Sub
Examples:

- AlignCurve
- ParcelCurve

Sub Example_TangentLength_AlignCurve()

' This example returns the TangentLength for the first Curve found in the first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Curve entity in the first Alignment."

' Find first Curve in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kCurve Then
        alignMsg = "The TangentLength for the first Curve in the alignment is: " & Format(alignEnt.TangentLength, "0.00")
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "TangentLength Example"

End Sub

Sub Example_TangentLength_ParcelCurve()

' This example returns the TangentLength for the first curve in the first Parcel in the collection.

Dim parcel As AeccParcel
Dim parcelEnt As AeccParcelEntity
Set parcel = AeccApplication.ActiveProject.Parcels.Item(0)
Set parcelEnt = parcel.ParcelEntities.Item(0)

Dim parcelMsg As String
parcelMsg = "There is no Curve entity in the first Parcel."

' Find first Curve in the parcel
For Each parcelEnt In parcel.ParcelEntities
    If parcelEnt.Type = kParcelCurve Then
        parcelMsg = "The TangentLength for the first Curve in the Parcel is: " _
                    & Format(parcelEnt.TangentLength, "0.00")
        Exit For
    End If
End If
Next

MsgBox parcelMsg, vbInformation, "TangentLength Example"

End Sub
TempPath Example

Sub Example_TempPath()

' This example returns the TempPath setting.
Dim prefFiles As AeccPreferencesFiles
Set prefFiles = AeccApplication.Preferences.Files

MsgBox "The current value for TempPath is: " & prefFiles.TempPath, vbInformation

End Sub
Sub Example_TextAbove()

' This example returns the TextAbove setting for the first CurveText object in a selection set

On Error Resume Next

' Delete existing SelectionSet
ThisDrawing.SelectionSets("SSet").Delete

' Create the selection set based on a point selection and filter for CurveText objects
Dim ssetObj As AcadSelectionSet
Set ssetObj = ThisDrawing.SelectionSets.Add("SSet")

Dim gpCode(0) As Integer
Dim dataValue(0) As Variant

gpCode(0) = 0
dataValue(0) = "AEC_CURVETEXT"

Dim groupCode As Variant
Dim dataCode As Variant

groupCode = gpCode
dataCode = dataValue
ssetObj.SelectOnScreen groupCode, dataCode

Dim objCrvTxt As AeccCurveText
Set objCrvTxt = ssetObj.Item(0)

MsgBox "The setting for TextAbove is: " & objCrvTxt.TextAbove, vbInformation

End Sub
Sub Example_TextBelow()

' This example returns the TextBelow setting for the
' first CurveText object in a selection set

On Error Resume Next

' Delete existing SelectionSet
ThisDrawing.SelectionSets("SSet").Delete

' Create the selection set based on a point selection
' and filter for CurveText objects
Dim ssetObj As AcadSelectionSet
Set ssetObj = ThisDrawing.SelectionSets.Add("SSet")

Dim gpCode(0) As Integer
Dim dataValue(0) As Variant

gpCode(0) = 0
dataValue(0) = "AEC_CURVETEXT"

Dim groupCode As Variant
Dim dataCode As Variant

groupCode = gpCode
dataCode = dataValue
ssetObj.SelectOnScreen groupCode, dataCode

Dim objCrvTxt As AeccCurveText
Set objCrvTxt = ssetObj.Item(0)

MsgBox "The setting for TextBelow is: " & objCrvTxt.TextBelow, vbInformation

End Sub
Sub Example_TextHeight()

    ' This example displays the TextHeight setting for the current drawing.
    Dim dbPref As AeccDatabasePreferences
    Set dbPref = AeccApplication.ActiveDocument.preferences

    MsgBox "The current value for TextHeight is: " & dbPref.TextHeight, _
           vbInformation, "TextHeight Example"

End Sub
Sub Example_TextLayer()

    ' This example returns the TextLayer for the first ProfileBlock
    ' in the collection
    Dim alignProf As AeccProfileBlock
    Set alignProf = AeccApplication.ActiveDocument.ProfileBlocks.Item(0)

    MsgBox "The TextLayer for the first ProfileBlock in the collection is: " _
        & alignProf.TextLayer, vbInformation, "TextLayer Example"

End Sub
Sub Example_TextOffsetAbove()

' This example returns the TextOffsetAbove setting for the first CurveText object in a selection set

On Error Resume Next

' Delete existing SelectionSet
ThisDrawing.SelectionSets("SSet").Delete

' Create the selection set based on a point selection and filter for CurveText objects
Dim ssetObj As AcadSelectionSet
Set ssetObj = ThisDrawing.SelectionSets.Add("SSet")

Dim gpCode(0) As Integer
Dim dataValue(0) As Variant

gpCode(0) = 0
dataValue(0) = "AEC_CURVETEXT"

Dim groupCode As Variant
Dim dataCode As Variant

groupCode = gpCode
dataCode = dataValue
ssetObj.SelectOnScreen groupCode, dataCode

Dim objCrvTxt As AeccCurveText
Set objCrvTxt = ssetObj.Item(0)

MsgBox "The setting for TextOffsetAbove is: " & objCrvTxt.TextOffsetAbove

End Sub
Sub Example_TextOffsetBelow()

' This example returns the TextOffsetBelow setting for the first CurveText object in a selection set

On Error Resume Next

' Delete existing SelectionSet
ThisDrawing.SelectionSets("SSet").Delete

' Create the selection set based on a point selection and filter for CurveText objects
Dim ssetObj As AcadSelectionSet
Set ssetObj = ThisDrawing.SelectionSets.Add("SSet")

Dim gpCode(0) As Integer
Dim dataValue(0) As Variant

gpCode(0) = 0
dataValue(0) = "AEC_CURVETEXT"

Dim groupCode As Variant
Dim dataCode As Variant

groupCode = gpCode
dataCode = dataValue
ssetObj.SelectOnScreen groupCode, dataCode

Dim objCrvTxt As AeccCurveText
Set objCrvTxt = ssetObj.Item(0)

MsgBox "The setting for TextOffsetBelow is: " & objCrvTxt.TextOffsetBelow

End Sub
Sub Example_TextSize()

' This example returns the TextSize setting for the first CurveText object in a selection set

On Error Resume Next

' Delete existing SelectionSet
ThisDrawing.SelectionSets("SSet").Delete

' Create the selection set based on a point selection
' and filter for CurveText objects
Dim ssetObj As AcadSelectionSet
Set ssetObj = ThisDrawing.SelectionSets.Add("SSet")

Dim gpCode(0) As Integer
Dim dataValue(0) As Variant

gpCode(0) = 0
dataValue(0) = "AEC_CURVETEXT"

Dim groupCode As Variant
Dim dataCode As Variant

groupCode = gpCode
dataCode = dataValue
ssetObj.SelectOnScreen groupCode, dataCode

Dim objCrvTxt As AeccCurveText
Set objCrvTxt = ssetObj.Item(0)

MsgBox "The setting for TextSize is: " & objCrvTxt.TextSize, vbInformation,

End Sub
Sub Example_ThetaExt()

' This example returns the ThetaExt for the first Spiral found in the first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Spiral entity in the first Alignment."

' Find first Spiral in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kSpiral Then
        alignMsg = "The ThetaExt for the first Spiral in the alignment is: " & alignEnt.ThetaExt
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "ThetaExt Example"

End Sub
Time Example

Sub Example_Time()

' This example returns the Time for the first FileLock in the collection.
Dim filelock As AeccFileLock
Set filelock = AeccApplication.ActiveProject.fileLocks.Item(0)

MsgBox "The Time of the first FileLock in the collection is: " & filelock.Time,
   vbInformation, "Time Example"

End Sub
TinPoints Example

Sub Example_TinPoints()

' This example returns the number of TinPoints in the first
' surface in the collection.
Dim surf As AeccSurface
Dim surfOut As AeccSurfaceOutputs
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set surfOut = surf.Outputs

MsgBox "The number of TinPoints in the first Surface is: " & surfOut.TinPoints.Count,
    vbInformation, "TinPoints Example"

End Sub
Sub Example_TotalX()

' This example returns the TotalX for the first Spiral found in the first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Spiral entity in the first Alignment."

' Find first Spiral in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kSpiral Then
        alignMsg = "The TotalX for the first Spiral in the alignment is: " & alignEnt.TotalX
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "TotalX Example"

End Sub
**TotalY Example**

Sub Example_TotalY()

' This example returns the TotalY for the first Spiral found in the
' first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnts As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)

Dim alignMsg As String
alignMsg = "There is no Spiral entity in the first Alignment."

' Find first Spiral in the alignment
For Each alignEnt In align.AlignEntities
    If alignEnt.Type = kSpiral Then
        alignMsg = "The TotalY for the first Spiral in the alignment is: " & alignE
        Exit For
    End If
Next

MsgBox alignMsg, vbInformation, "TotalY Example"

End Sub
Type Example

Examples:

- AlignEntity, AlignCurve, AlignCurve, AlignTangent
- Boundary
- Breakline
- CrossSectionSurface (Civil Engineering Feature)
- EGProfile (Civil Engineering Feature)
- FGProfile (Civil Engineering Feature)
- ParcelEntity, ParcelCurve, ParcelLine
- StationEquation
- Surface
- Watershed

Sub Example_Type_AlignEntity()

' This example returns the Type setting for the first entity in the first Alignment in the collection.
Dim align As AeccAlignment
Dim alignEnt As AeccAlignEntity
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set alignEnt = align.AlignEntities.Item(0)

Dim alignType As String

Select Case alignEnt.Type
Case kCurve
    alignType = "Curve"
Case kSpiral
    alignType = "Spiral"
Case kTangent
    alignType = "Tangent"
End Select

MsgBox "The Type for the first entity in the alignment is: " & alignType _
, vbInformation, "Type Example"

End Sub

Sub Example_Type_Boundary()

' This example returns the Type ' for the first Boundary in the collection.
Dim surf As AeccSurface
Dim bound As AeccBoundary
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set bound = surf.Inputs.Boundaries.Item(0)

Dim bndType As String

Select Case bound.Type
Case kBoundaryTypeShow
    bndType = "Visible"
Case kBoundaryTypeHide
    bndType = "Hidden"
Case kBoundaryTypeOuter
    bndType = "Outer"
End Select

MsgBox "The Type for the first Boundary is: " & bndType _
, vbInformation, "Type Example"

End Sub

Sub Example_Type_BreakLine()
'This example returns the Type
' for the first BreakLine in the collection.
Dim surf As AeccSurface
Dim brkLine As AeccBreakLine
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set brkLine = surf.Inputs.BreakLines.Item(0)

Dim brkType As String

Select Case brkLine.Type
Case kStandard
  brkType = "Standard"
Case kProximity
  brkType = "Proximity"
Case kWallLeft
  brkType = "Wall Left"
Case kWallRight
  brkType = "Wall Right"
Case kNondestructive
  brkType = "Non Destructive"
End Select

MsgBox "The Type for the first BreakLine is: " & brkType _, vbInformation, "Type Example"

End Sub

Sub Example_Type_CrossSectionSurface()

  ' This example returns the surface type for the first cross
  ' section surface in the collection for the first cross section
  ' in the collection.
  Dim aligns As AeccAlignments
  Dim align As AeccAlignment
  Dim xSect As AeccCrossSection
  Dim xSectSurf As AeccCrossSectionSurface
Set aligns = AeccApplication.ActiveProject.Alignments
Set align = aligns.Item(0)
Set xSect = align.CrossSections.Item(0)
Set xSectSurf = xSect.CrossSectionSurfaces.Item(0)

' Get the alignment name
Dim alignName As String
alignName = align.Name

' Get the cross section station and format it
Dim station As String
station = aligns.DoubleToStaFormat(xSect.station)

' Get the cross section surface name
Dim surfName As String
surfName = xSectSurf.Name

' Get the cross section surface type
Dim xSurfType As String

Select Case xSectSurf.Type
Case kExistingGround
    xSurfType = "Existing Ground"
Case kTopSurface
    xSurfType = "Top Surface"
Case kDatumSurface
    xSurfType = "Datum Surface"
Case kTemplateSurface
    xSurfType = "Template Surface"
Case kSubassemblySurface
    xSurfType = "Subassembly Surface"
Case kMatchSurface
    xSurfType = "Match Surface"
End Select

MsgBox "The alignment name is: " & alignName & vbCrLf & 
    "The first cross section is at station: " & station & vbCrLf & 
    "The cross section surface type is: " & xSurfType
Sub Example_Type_EGProfile()

' This example returns the Type setting for the first existing ground profile
' in the first alignment in the collection.
Dim align As AeccAlignment
Dim EGProf As AeccEGProfile
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set EGProf = align.EGProfiles.Item(0)

Dim EGProfType As String

Select Case EGProf.Type
Case kEgCenter
    EGProfType = "Center"
Case kEgLeft
    EGProfType = "Left"
Case kEgNone
    EGProfType = "None"
Case kEgRight
    EGProfType = "Right"
End Select

MsgBox "The Type for the first existing ground profile in the alignment is: " & EGProfType & vbCrLf & _
    "The first surface type is: " & xSurfType, vbInformation, "Type Example"

End Sub

Sub Example_Type_FGProfile()

' This example returns the Type setting for the finished ground profile
' in the first alignment in the collection.
Dim align As AeccAlignment
Dim FGProf As AeccFGProfile
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set FGProf = align.FGProfiles.Item(0)

Dim FGProfType As String

Select Case FGProf.Type
    Case kFgCenter
        FGProfType = "Center"
    Case kFgDitchLeft
        FGProfType = "Ditch Left"
    Case kFgDitchRight
        FGProfType = "Ditch Right"
    Case kFgLeft1
        FGProfType = "Left 1"
    Case kFgLeft2
        FGProfType = "Left 2"
    Case kFgLeft3
        FGProfType = "Left 3"
    Case kFgLeft4
        FGProfType = "Left 4"
    Case kFgLeft5
        FGProfType = "Left 5"
    Case kFgLeft6
        FGProfType = "Left 6"
    Case kFgLeft7
        FGProfType = "Left 7"
    Case kFgLeft8
        FGProfType = "Left 9"
    Case kFgNone
        FGProfType = "None"
    Case kFgRight1
        FGProfType = "Right 1"
    Case kFgRight2
        FGProfType = "Right 2"
    Case kFgRight3
        FGProfType = "Right 3"
Case kFgRight4
    FGProfType = "Right 4"
Case kFgRight5
    FGProfType = "Right 5"
Case kFgRight6
    FGProfType = "Right 6"
Case kFgRight7
    FGProfType = "Right 7"
Case kFgRight8
    FGProfType = "Right 8"
End Select

MsgBox "The Type for the finished ground profile in the alignment is: " & FGProfType, vbInformation, "Type Example"

End Sub

Sub Example_Type_ParcelEntity()

    ' This example returns the Type for the first entity in the first Parcel in the collection.
    Dim parcel As AeccParcel
    Dim parcelEnt As AeccParcelEntity
    Set parcel = AeccApplication.ActiveProject.Parcels.Item(0)
    Set parcelEnt = parcel.ParcelEntities.Item(0)

    Dim parcelType As String

    Select Case parcelEnt.Type
        Case kParcelCurve
            parcelType = "Curve"
        Case kParcelLine
            parcelType = "Line"
    End Select

    MsgBox "The Type of the first entity in the Parcel is: " & parcelType, vbInformation, "Type Example"
Sub Example_Type_StationEquation()

' This example returns the Type setting for StationEquation.
Dim align As AeccAlignment
Dim staEqu As AeccStationEquation
Set align = AeccApplication.ActiveProject.Alignments.Item(0)
Set staEqu = align.StationEquations.Item(0)

' Convert the type to a string.
Select Case staEqu.Type
Case kIncreasing
   staType = "Station Equation increases stationing."
Case kDecreasing
   staType = "Station Equation decreases stationing."
End Select

MsgBox "The current value for Type is " & staType, vbInformation, "Type Example"

End Sub

Sub Example_Type_Surface()

' This example returns the Type for the first surface in the collection.
Dim surf As AeccSurface
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

Dim surfType as String

' Convert the type to a string.
Select Case surf.Type
Case kUnknownSurface
   surfType = "The surface type is Unknown."
Case kTin
   surfType = "The surface type is Surface."
End Select

End Sub
Case kCompositeVolumn
    surfType = "The surface type is Composite Volume."
Case kGridVolumn
    surfType = "The surface type is Grid Volume."
End Select

MsgBox surfType, vbInformation, "Type Example"

End Sub

Sub Example_Type_Watershed()

    ' This example returns the Type or Watershed
    ' for the first WaterShed in the collection.
    Dim surf As AeccSurface
    Dim wShed As AeccWaterShed
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
    Set wShed = surf.Outputs.WaterSheds.Item(0)

    Dim wShedType As String

    Select Case wShed.Type
    Case kBoundaryPoint
        wShedType = "includes a boundary point."
    Case kBoundarySegment
        wShedType = "includes part of a boundary."
    Case kLocalMin
        wShedType = "is a depression."
    Case kFlatArea
        wShedType = "has a flat area."
    Case kMultiDrain
        wShedType = "has multiple drains."
    End Select

    MsgBox "The first WaterShed " & wShedType, vbInformation, "Type Example"
End Sub
Sub Example_UnLockPoints()

' This examples locks a series of point numbers and
' then unlocks a subset of those point numbers.
Dim cogoPnts As AeccCogoPoints
Set cogoPnts = AeccApplication.ActiveProject.CogoPoints

' Create locked point string
Dim pntString As String
pntString = "1,4-8,40,10,12"

' Lock points
cogoPnts.LockPoints (pntString)

MsgBox "Locked point numbers are " & cogoPnts.LockedPointNumbers, _
vbInformation, "UnLockPoints Example"

' Create unlock point string
pntString = "4-8"

' Unlock points
cogoPnts.UnlockPoints (pntString)

MsgBox "Locked point numbers are " & cogoPnts.LockedPointNumbers, _
vbInformation, "UnLockPoints Example"

End Sub
Sub Example_UpperRight()

' This example returns the Datum Elevation for the first ProfileBlock ' in the collection.
Dim alignProf As AeccProfileBlock
Set alignProf = AeccApplication.ActiveDocument.ProfileBlocks.Item(0)

Dim coords As Variant
coords = alignProf.UpperRight

MsgBox "The UpperRight coordinates for the first Alignment Profile are:" & vbCrLf & 
"X Value: " & Format(coords(0), "0.00") & vbCrLf & 
"Y Value: " & Format(coords(1), "0.00"), vbInformation, "UpperRight Exampl

End Sub
UsedPointNumbers Example

Sub Example_UsedPointNumbers()

' This gets the UsedPointNumbers in the CogoPoints collection
Dim cogoPnts As AeccCogoPoints
Set cogoPnts = AeccApplication.ActiveProject.CogoPoints

MsgBox "The UsedPointNumbers in the CogoPoints collection are: " & cogoPnts
vbInformation, "UsedPointNumbers Example"

End Sub
Sub Example_User()

' This example returns the current setting of LastUsedProject
' from the preferences object.
Dim prefs As AeccPreferences
Set prefs = AeccApplication.Preferences

MsgBox "The last used project was " & prefs.User.LastUsedProj, vbInformatic

End Sub
Sub Example_Utility()

' This example returns the AngleToString value for 0.785398 radians using
' the active document.
Dim doc As AeccDocument
Set doc = AeccApplication.ActiveDocument

MsgBox "The setting for AngleToString is " & doc.Utility.AngleToString(0.785398,
, vbInformation, "Utility Example"

End Sub
Sub Example_VerticalGridLayer()

' This example returns the VerticalGridLayer for the first ProfileBlock
' in the collection
Dim alignProf As AeccProfileBlock
Set alignProf = AeccApplication.ActiveDocument.ProfileBlocks.Item(0)

MsgBox "The VerticalGridLayer for the first ProfileBlock in the collection is: 
& alignProf.VerticalGridLayer, vbInformation, "VerticalGridLayer Example"

End Sub
VerticalScale Example

Examples:

- **CrossSectionBlock** (Civil Engineering Feature)
- **DatabasePreferences**
- **ProfileBlock** (Civil Engineering Feature)

---

Sub Example_VerticalScale_CrossSectionBlock()

' This example returns the vertical scale for the first alignment cross section in the collection.
Dim alignXsects As AeccCrossSectionBlocks
Dim alignXsect As AeccCrossSectionBlock
Set alignXsects = AeccApplication.ActiveDocument.CrossSectionBlocks
Set alignXsect = alignXsects.Item(0)

' Get the station for the first alignment cross section in the collection
Dim station As String
station = alignXsect.station

MsgBox "The vertical scale for the alignment cross section at station " & station & ", is: " & Format(alignXsect.VerticalScale, "0.00"), vbInformation, "VerticalScale Example"

End Sub

---

Sub Example_VerticalScale_DatabasePreferences()

' This example returns the vertical scale setting for the current drawing.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.Preferences

MsgBox "The current vertical scale for this drawing is: " & dbPref.VerticalScale, vbInformation, "VerticalScale Example"
Sub Example_VerticalScale_ProfileBlock()

    ' This example returns the vertical scale for the first alignment profile
    ' in the collection.
    Dim alignProf As AeccProfileBlock
    Set alignProf = AeccApplication.ActiveDocument.ProfileBlocks.Item(0)

    MsgBox "The vertical scale for the first alignment profile is: " & _
    Format(alignProf.VerticalScale, "0.00"), vbInformation, "VerticalScale Exam"
Sub Example_Volume()

    ' This example returns the Volume for the first surface in the collection.
    Dim surf As AeccSurface
    Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)

    MsgBox "The Volume for the first surface is: " & surf.Volume, vbInformation,

End Sub
Sub Example_VolumeDisplayUnit()

' This example returns the VolumeDisplayUnit setting for the current drawing.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.preferences

' Convert the volume display unit to a string.
Dim unit As String
Select Case dbPref.VolumeDisplayUnit
  Case aecUnitCubicInch
    unit = "cubic inch"
  Case aecUnitCubicFoot
    unit = "cubic foot"
  Case aecUnitCubicYard
    unit = "cubic yard"
  Case aecUnitCubicMil
    unit = "cubic millimeters"
  Case aecUnitCubicCentimeter
    unit = "cubic centimeters"
  Case aecUnitCubicDecimeter
    unit = "cubic decimeters"
  Case aecUnitCubicMeter
    unit = "cubic meters"
End Select

MsgBox "The current value for VolumeDisplayUnit is " & unit, _
  vbInformation, "VolumnDisplayUnit Example"
End Sub
VolumnPrecision Example

Sub Example_VolumePrecision()

' This example displays the VolumePrecision setting for the current drawing.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.preferences

MsgBox "The current value for VolumePrecision is: " & dbPref.VolumePrecision, vbInformation, "VolumnPrecision Example"

End Sub
VolumnSuffix Example

Sub Example_VolumeSuffix()

' This example displays the VolumeSuffix setting for the current drawing.
Dim dbPref As AeccDatabasePreferences
Set dbPref = AeccApplication.ActiveDocument.preferences

MsgBox "The current value for VolumeSuffix is " & dbPref.VolumeSuffix, _
        vbInformation, "VolumnSuffix Example"

End Sub
Sub Example_Watersheds()

' This example returns the number of WaterSheds in the first
' surface in the collection.
Dim surf As AeccSurface
Dim surfOut As AeccSurfaceOutputs
Set surf = AeccApplication.ActiveProject.Surfaces.Item(0)
Set surfOut = surf.Outputs

MsgBox "The number of WaterSheds in the first Surface is: " & surfOut.WaterSheds.Count, vbInformation, "WaterSheds Example"

End Sub
Sub Example_XyToEastNorth()

' This example returns the Easting and Northing for an AutoCAD XY.
Dim util As AeccUtility
Set util = AeccApplication.ActiveDocument.Utility

Dim ptXY (0 To 2) As Double
Dim ptEN(0 To 2) As Double
Dim var1 As Variant
Dim str As String

ptXY(0) = 1000
ptXY(1) = 1000
ptXY(2) = 0

' Convert point to Easting, Northing
var1 = util.XyToEastNorth(ptXY)

ptEN(0) = var1(0)
ptEN(1) = var1(1)
ptEN(2) = var1(2)

' Make display string
str = Format(ptEN(0), "0.00") + ", " + Format(ptEN(1), "0.00") + ", " + Format(ptEN(2), "0.00")
MsgBox "Point: " & str, vbInformation, "XyToEastNorth Example"

End Sub
Sub Example_XyToOffsetElevation()

' This example returns the offset and elevation for a selected point in the first
' alignment crosssection in the collection
Dim alignXSec As AeccCrossSectionBlock
Set alignXSec = AeccApplication.ActiveDocument.CrossSectionBlocks.Item(0)

Dim station As String
Dim offElev As Variant
Dim returnPnt As Variant

' Get the station for the first alignment cross section in the collection.
station = alignXSec.station

' Get a point in the alignment cross section.
returnPnt = ThisDrawing.Utility.GetPoint(vbCrLf & "Specify a point in the al

offElev = alignXSec.XyToOffsetElevation(returnPnt)

MsgBox "The station for the selected point is: " & Format(offElev(0), "0.00") & vbCrLf & "The offset for the selected point is: " & Format(offElev(1), "0.00") & vbCrLf, vbInformation, "XyToOffsetElevation Example"

End Sub
Sub Example_XyToStationElevation()

' This example returns the XyToStationElevation for the first ProfileBlock
' in the collection
Dim alignProf As AeccProfileBlock
Set alignProf = AeccApplication.ActiveDocument.ProfileBlocks.Item(0)

Dim staElev As Variant
Dim returnPnt As Variant

' Get a point in the alignment profile.
returnPnt = ThisDrawing.Utility.GetPoint(, "Enter a point in the alignment prof"
staElev = alignProf.XyToStationElevation(returnPnt)

MsgBox "The Station for the X value is: " & Format(staElev(0), "0.00") & vbCrLf & "The Elevation for the Y Value is: " & Format(staElev(1), "0.00") , vbInformation, "XyToStationElevation Example"

End Sub
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding the ActiveX Object Model
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding the ActiveX Object Model
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding COM Wrappers
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Parcels
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

Status

ActiveX and VBA Developer’s Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding the ActiveX Object Model
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Profiles
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

- **FindPoint**
- **GetElevation**

ActiveX and VBA Developer's Guide:

- **Understanding Surfaces**
See Also

Methods and Properties:

SearchType

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

SetBoundingBox

Build

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

FindPoint

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Cross Sections and Superelevations

Understanding Profiles
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
Understanding Cross Sections and Superelevations
Understanding Parcels
Understanding Profiles
Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Profiles
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding the ActiveX Object Model
Collections

AlignEntities
Alignments
Boundaries
BreakLines
CogoPoints
ContourItems
CrossSectionBlocks (Civil Engineering Feature)
CrossSectionPointCodes (Civil Engineering Feature)
CrossSections (Civil Engineering Feature)
CrossSectionSurfaces (Civil Engineering Feature)
DEMFiles
DescriptionKeyFile
DescriptionKeyFiles
ElevationContours
Faces
FGProfiles (Civil Engineering Feature)
FileLocks
Parcels
ParcelEntities
PointFiles
PointGroups
PointGroupNames
ProfileBlocks (Civil Engineering Feature)
Projects
PVIs (Civil Engineering Feature)
Prototypes
StationEquations
Documents

Drawings

Edges

EGProfiles (Civil Engineering Feature)

ElevationContours

SuperElevations (Civil Engineering Feature)

Surfaces

TinPoints

WaterSheds
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

LockedPointNumbers

ActiveX and VBA Developer’s Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Cross Sections and Superelevations
See Also

Methods and Properties:

Save

ActiveX and VBA Developer’s Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Cross Sections and Superelevations
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Profiles
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding COM Wrappers
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

[Understanding COM Wrappers](#)
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

AutoSave

Open

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

[Understanding Project Management](#)
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Cross Sections and Superelevations
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

[Understanding Alignments](#)

[Understanding Cross Sections and Superelevations](#)
See Also

Methods and Properties:

GetBoundingBox

ActiveX and VBA Developer’s Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding COM Wrappers
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Profiles
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

LockedPointNumbers

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

[Understanding Project Management]
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Cross Sections and Superelevations
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Profiles
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

[Understanding Project Management]
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding the ActiveX Object Model
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Parcels
See Also

Methods and Properties:

Area3D
AverageGrade
MaxFaceArea
MaxGrade
MeanElevation
MinFaceArea
MinGrade
NumberOfFaces
Volume

ActiveX and VBA Developer's Guide:
Understanding Surfaces
See Also

Methods and Properties:

Area2D
AverageGrade
MaxFaceArea
MaxGrade
MeanElevation
MinFaceArea
MinGrade
NumberOfFaces
Volume

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

Save

ActiveX and VBA Developer’s Guide:

Understanding Project Management
See Also

Methods and Properties:

Save

ActiveX and VBA Developer’s Guide:

Understanding Alignments

Understanding CogoPoints
See Also

Methods and Properties:

Area2D
Area3D
MaxFaceArea
MaxGrade
MeanElevation
MinFaceArea
MinGrade
NumberOfFaces
Volume

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

[Understanding Project Management]
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

[Understanding Surfaces]
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Parcels
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Cross Sections and Superelevations
Sub Example_CenterlineOffset()

    ' This example returns the centerline offset for the
    ' first alignment cross section in the collection.
    Dim alignXsects As AeccCrossSectionBlocks
    Dim alignXsect As AeccCrossSectionBlock
    Set alignXsects = AeccApplication.ActiveDocument.CrossSectionBlocks
    Set alignXsect = alignXsects.Item(0)

    'Get the station for the first alignment cross section in the collection
    Dim station As String
    station = alignXsect.station

    MsgBox "The centerline offset for the alignment cross section at station " & station & ":" & Format(alignXsect.CenterlineOffset, "0.00"), vbInformation, "Height Example"

End Sub
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Parcels
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Parcels
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Parcels
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints

Understanding Cross Sections and Superelevations
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding COM Wrappers
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding COM Wrappers
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints

Understanding COM Wrappers

Understanding Cross Sections and Superelevations

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding the ActiveX Object Model
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Cross Sections and Superelevations
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Parcels
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Parcels
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Cross Sections and Superelevations
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Cross Sections and Superelevations
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Cross Sections and Superelevations
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Cross Sections and Superelevations
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Cross Sections and Superelevations
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Cross Sections and Superelevations
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

[Understanding Parcels](#)
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Profiles
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Cross Sections and Superelevations

Understanding Profiles
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Parcels
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

Name

ActiveX and VBA Developer's Guide:

Understanding Alignments
Understanding CogoPoints
Understanding COM Wrappers
Understanding Cross Sections and Superelevations
Understanding Project Management
Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

[Understanding Project Management]
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Profiles
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
Understanding COM Wrappers
Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Profiles
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Profiles
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
Understanding CogoPoints
Understanding COM Wrappers
Understanding Cross Sections and Superelevations
Understanding Profiles
Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

[Understanding Alignments]
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Parcels
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Profiles
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Parcels
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Profiles
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Profiles
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Cross Sections and Superelevations
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

PointNameSize

ActiveX and VBA Developer’s Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding COM Wrappers
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

[Understanding Project Management]
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Cross Sections and Superelevations
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Parcels
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

[Understanding Alignments](#)
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

LockPoints

UnlockPoints

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
Understanding CogoPoints
Understanding Project Management
Understanding Surfaces
See Also

Methods and Properties:

*SpiralType2*

ActiveX and VBA Developer's Guide:

*Understanding Alignments*
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Cross Sections and Superelevations

Understanding Surfaces
See Also

Methods and Properties:

Area2D
Area3D
AverageGrade
MaxGrade
MeanElevation
MinFaceArea
MinGrade
NumberOfFaces
Volume

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

Area2D
Area3D
AverageGrade
MaxFaceArea
MeanElevation
MinFaceArea
MinGrade
NumberOfFaces
Volume

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Cross Sections and Superelevations
See Also

Methods and Properties:

Area2D
Area3D
AverageGrade
MaxFaceArea
MaxGrade
MinFaceArea
MinGrade
NumberOfFaces
Volume

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

[Understanding Surfaces](#)
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
Understanding Cross Sections and Superelevations
Understanding Surfaces
See Also

Methods and Properties:

Area2D
Area3D
AverageGrade
MaxFaceArea
MaxGrade
MeanElevation
MinGrade
NumberOfFaces
Volume

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

Area2D
Area3D
AverageGrade
MaxFaceArea
MaxGrade
MeanElevation
MinFaceArea
NumberOfFaces
Volume

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Cross Sections and Superelevations
See Also

Methods and Properties:

Description

ActiveX and VBA Developer's Guide:

Understanding Alignments
Understanding CogoPoints
Understanding COM Wrappers
Understanding Cross Sections and Superelevations
Understanding Parcels
Understanding Profiles
Understanding Project Management
Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints

Understanding COM Wrappers

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
Understanding CogoPoints
Understanding COM Wrappers
Understanding Parcels
See Also

Methods and Properties:

Area2D
Area3D
AverageGrade
MaxFaceArea
MaxGrade
MeanElevation
MinFaceArea
MinGrade
Volume

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

NumberofFaces

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Cross Sections and Superelevations
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:
  Understanding Alignments
  Understanding Cross Sections and Superelevations
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

[Understanding CogoPoints]
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

[Understanding Project Management]
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Parcels
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Parcels
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

[Understanding Parcels]
Sub Example_Parcsels()

  ' This example returns the number of Parcsels in the current project.
  Dim proj As AeccProject
  Set proj = AeccApplication.ActiveProject

  MsgBox "The number of Parcsels in the current Project is: " & proj.Parcels.Count, vbInformation, "Parcsels Example"

End Sub
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Parcels
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

_isNameSupported_

ActiveX and VBA Developer’s Guide:

_Understanding CogoPoints_
See Also

Methods and Properties:

PointTolerance

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

PointOnLineTolerance

ActiveX and VBA Developer’s Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

[Understand Parcels](#)
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

[Understanding Project Management](#)
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Profiles
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Profiles
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Profiles
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

[Understanding CogoPoints](#)
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Cross Sections and Superelevations
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

[Understanding CogoPoints](#)
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

[Understanding CogoPoints](#)
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

[Understanding CogoPoints]
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

FindPath

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

[Understanding Project Management](#)
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

SpiralType2

ActiveX and VBA Developer’s Guide:

Understanding Alignments
See Also

Methods and Properties:

LOffset

SpiralType1

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Profiles
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
Understanding Cross Sections and Superelevations
Understanding Profiles
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Profiles
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Profiles
See Also

Methods and Properties:

Build

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Cross Sections and Superelevations
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Profiles
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Profiles
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding COM Wrappers
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding COM Wrappers
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding COM Wrappers
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding COM Wrappers
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding COM Wrappers
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments

Understanding Profiles
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding CogoPoints
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Alignments
Understanding Cross Sections and Superelevations
Understanding Profiles
Understanding Project Management
See Also

Methods and Properties:

Area2D
Area3D
AverageGrade
MaxGrade
MaxFaceArea
MeanElevation
MinFaceArea
MinGrade
NumberOfFaces

ActiveX and VBA Developer's Guide:

Understanding Surfaces
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Project Management
See Also

Methods and Properties:

ActiveX and VBA Developer's Guide:

Understanding Surfaces