Objects

**Aec2dSection** AEC 2D section object.

**AecAnchor** Base class for all AEC anchor objects.

**AecAnchorEntToCurve** Binds one AEC object to another AEC or AutoCAD object, such as a roof, wall, line, or arc.

**AecAnchorEntToGridAssembly** Attaches entity objects, such as doors and windows, to grid assemblies, such as curtain walls and window assemblies.

**AecAnchorEntToLayoutCell** Attaches objects, such as light fixtures, to cells on a 2D layout grid.

**AecAnchorEntToLayoutNode** Attaches objects, such as columns, to nodes on a layout grid.

**AecAnchorEntToLayoutVolume** Attaches objects, such as furniture, to volumes on a 3D layout grid.

**AecAnchorLeadEntToNode** Connects an object to a layout node with a leader between the two objects.

**AecAnchorToRef** Base class for referencing objects from an anchor.

**AecBaseApplication** Application object extending AcadApplication to provide support for AEC Applications.

**AecBaseDatabase** AEC database object extending AcadDatabase to provide support for general AEC objects.

**AecBaseDatabasePreferences** A derivative of AecDatabasePreferences, providing base desktop database preferences support.

**AecBaseDocument** An AEC drawing object, extending the AcadDocument to provide support for general AEC objects.

**AecBaseDocuments** The collection of all AEC drawings open in the
current session.

AecBasePreferences AEC AutoCAD preferences object.

AecBlockRef An instance of an AEC block in a drawing.

AecCamera AEC camera object.

AecCellLayoutTool Base Class for the layout grid objects.

AecClipVol Clip Volume entity defining an extruded region for clipping.

AecClipVolRes Clip Volume Result which captures the resulting graphics for a section or elevation for insertion into the drawing.

AecDatabase AEC Database object extending the Acad Database object.

AecDatabasePreferences Extends the AutoCAD Database Preferences object to include AEC properties and methods.

AecDictionary AEC Dictionary containing a collection of AEC objects.

AecDictRecord An AEC Dictionary item.

AecEditInPlaceProfile In-place edit profile object.

AecEntity Base class for all AEC entities.

AecEntRef A reference to any entity in the drawing, similar to a block reference.

AecGeo Provides location, rotation and anchoring capabilities for AEC entities.

AecGridAssembly Base class for grid assemblies.

AecLayerKey Layer key definition, which specifies the layer properties generated from a particular key.

AecLayerKeys The collection of layer key definitions in a layer key style.
**AecLayerKeyStyle** A collection of layer keys.

**AecLayerKeyStyles** The collection of layer key styles in the specified drawing.

**AecLayerOverrideSetting** A layer key style override setting.

**AecLayerOverrideSettings** The collection of override settings in a layer key style.

**AecLayoutCurve** A single dimensional grid with nodes along a single curve.

**AecLayoutCurveNode** A node in a layout curve.

**AecLayoutCurveNodes** The collection of nodes in a layout curve.

**AecLayoutGrid2D** Layout grid entity used to layout any AEC object within a 2D grid.

**AecLayoutGrid2DNode** A node in a 2D layout grid.

**AecLayoutGrid2DNodes** The collection of nodes in a 2D layout grid.

**AecLayoutGrid2DXNode** A node along the X axis of a 2D layout grid.

**AecLayoutGrid2DXNodes** Collection of nodes along the X axis of a 2D layout grid.

**AecLayoutGrid2DYNode** A node along the Y axis of a 2D layout grid.

**AecLayoutGrid2DYNodes** Collection of nodes along the Y axis of a 2D layout grid.

**AecLayoutGrid3D** A layout volume for arranging objects in a 3D grid.

**AecLayoutGrid3DNode** A node of a 3D Layout Grid.

**AecLayoutGrid3DNodes** The collection of nodes in a 3D layout grid.

**AecLayoutGrid3DXNode** A node along the X axis of a layout volume.
**AecLayoutGrid3DXNodes** Collection of nodes along the X axis of a layout volume.

**AecLayoutGrid3DYNODE** A node along the Y axis of a layout volume.

**AecLayoutGrid3DYNodes** Collection of nodes along the Y axis of a layout volume.

**AecLayoutGrid3DZNNode** A node along the Z axis of a layout volume.

**AecLayoutGrid3DZNodes** Collection of nodes along the Z axis of a layout volume.

**AecLayoutNode** Provides common properties shared by all types of layout nodes.

**AecLayoutNodes** Collection of layout nodes.

**AecLayoutTool** Provides common functionality for layout curve and layout grid objects.

**AecMaskBlockRef** An instance of a mask block, a two-dimensional block that covers part of an AEC object in a plan (2D) view.

**AecMaskBlockStyle** Defines the properties of a mask block object.

**AecMaskBlockStyles** The collection of mask block styles in the specified drawing.

**AecMassElement** An object that has behaviors based on its shape.

**AecMassElementStyle** Defines the properties of a mass element object.

**AecMassElementStyles** The collection of mass element styles in the specified drawing.

**AecMassGroup** A group of mass elements combined to form a complex shape.

**AecMassGroups** The collection of mass groups in the specified drawing.
**AecMVBlockRef** An instance of a block that can have different representations in different view directions.

**AecMVBlockStyle** Defines the properties of a multi-view block object.

**AecMVBlockStyles** The collection of multi-view block styles in the specified drawing.

**AecObject** Provides common functionality for AEC database objects.

**AecPolygon** An Aec Polygon object.

**AecPolygonStyle** Defines the properties of an Aec Polygon object.

**AecPolygonStyles** The collection of Aec Polygon styles in the specified drawing.

**AecProfile** Defines the properties of a profile style.

**AecProfileStyle** A custom shape defined by two-dimensional closed polylines.

**AecProfileStyles** The collection of profile styles in the specified drawing.

**AecRing** A closed loop of polyline segments, often used as a group inside an AecProfile object.

**AecRings** The collection of rings in an AecProfile object.

**AecSlice** Slice of a massing group or any AEC object, typically used to generate floor plates and space boundaries.

**AecViewBlock** View block used for display representations by a multi-view block.

**AecViewBlocks** A collection of view blocks used for display representations by a multi-view block.

**AecVolumeLayoutTool** Provides standard functionality for all 3D layout tools.
AEC Base Object Model

Click on an object to view the description.
AEC 2D section object.

**VBA object name:** Aec2dSection

**Create using:** AddCustomObject("Aec2dSection")

**Access via:**
- ModelSpace.Item
- PaperSpace.Item
- Block.Item
- SelectionSet.Item
- Group.Item

**Methods**
- ArrayPolar
- ArrayRectangular
- AttachAnchor
- Copy

**Properties**
- Application
- ClipVol
- Description
Anchor Object

Base class for all AEC anchor objects.

**VBA object name:** AecAnchor

**Create using:** New AecAnchor

**Access via:** AecGeo.GetAnchor

You can use AecAnchor as a generic reference to anchor objects.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>AttachEntity</td>
</tr>
<tr>
<td>Description</td>
<td>Delete</td>
</tr>
<tr>
<td>Document</td>
<td>GetExtensionDictionary</td>
</tr>
<tr>
<td>Handle</td>
<td>Modified</td>
</tr>
<tr>
<td>HasExtensionDictionary</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Parameter</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>GetXData</td>
<td>ObjectID</td>
</tr>
<tr>
<td>SetXData</td>
<td>ObjectName</td>
</tr>
<tr>
<td></td>
<td>OwnerID</td>
</tr>
</tbody>
</table>
Anchor Entity to Curve

Binds one AEC object to another AEC or AutoCAD object, such as a roof, wall, line, or arc.

**VBA object name:** AecAnchorEntToCurve

**Create using:** New AecAnchorEntToCurve

**Access via:** AecGeo.GetAnchor

<table>
<thead>
<tr>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Document</td>
</tr>
<tr>
<td>FlipX</td>
</tr>
<tr>
<td>FlinY</td>
</tr>
</tbody>
</table>
Methods

AttachEntity
Delete
GetXData
SetXData

Flip
FlipZ
Handle
HasExtensionDictionary

ObjectID
ObjectName
OwnerID
Reference

GetXDistance
XPositionFrom
XPositionTo
XRotation
YDistance
YPositionFrom
YPositionTo

YRotation
ZDistance
ZPositionFrom
ZPositionTo
ZRotation
Object

Binds one AEC object to another.

**VBA object name:** AecAnchorEntToEnt

**Create using:** New AecAnchorEntToEnt

**Access via:** AecGeo.GetAnchor

### Properties

- **Application**
- **Description**
- **Document**
- **Handle**

### Methods

- **AttachEntity**
- **Delete**

### Events

- **GetExtensionDictionary**
- **HasExtensionDictionary**
- **Modified**
<table>
<thead>
<tr>
<th>Function</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetXData</td>
<td>ObjectId</td>
</tr>
<tr>
<td>SetXData</td>
<td>ObjectName</td>
</tr>
<tr>
<td></td>
<td>OwnerID</td>
</tr>
<tr>
<td></td>
<td>Reference</td>
</tr>
</tbody>
</table>
Assembly Object

Attaches entity objects, such as doors and windows, to grid assemblies, such as curtain walls and window assemblies.

VBA object name: AecAnchorEntToGridAssembly

Create using: N/A

Access via: AecGeo.GetAnchor

Properties

AdjustSizing
AllowVariation
Application
BottomOffset
Cell
<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
<th>Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttachEntity</td>
<td>FlipX</td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td>FlipY</td>
<td></td>
</tr>
<tr>
<td>GetExtensionDictionary</td>
<td>Handle</td>
<td></td>
</tr>
<tr>
<td>GetXData</td>
<td>FlipZ</td>
<td></td>
</tr>
<tr>
<td>SetXData</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HasExtensionDictionary</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LeftOffset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ObjectId</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ObjectName</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OwnerID</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reference</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RightOffset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TopOffset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>YAlignment</td>
<td></td>
</tr>
<tr>
<td></td>
<td>YOffset</td>
<td></td>
</tr>
</tbody>
</table>

**Events**

Modified
Attaches objects to cells on a 2D layout grid.

**VBA object name:** AecAnchorEntToLayoutCell

**Create using:** New AecAnchorEntToLayoutCell

**Access via:** AecGeo.GetAnchor

**Properties**
- Application
- ApplyResize
- CenterOnNode
- Description
- Document
Methods

- AttachEntity
- Delete
- GetExtensionDictionary
- GetXData
- SetXData
- FlipX
- FlipY
- FlipZ
- Handle
- HasExtensionDictionary
- Node
- Modified
- Events
- ObjectID
- ObjectName
- Offset
- OwnerID
- Reference
- ResizeOffset
- UseNodeCS
- XRotation
- YRotation
- ZRotation
Attaches objects, such as columns, to nodes on a layout grid.

**VBA object name:** AecAnchorEntToLayoutNode

**Create using:** New AecAnchorEntToLayoutNode

**Access via:** AecGeo.GetAnchor

<table>
<thead>
<tr>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
</tr>
<tr>
<td>CenterOnNode</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Document</td>
</tr>
<tr>
<td>FlipX</td>
</tr>
</tbody>
</table>
Methods

<table>
<thead>
<tr>
<th>AttachEntity</th>
<th>FlipY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete</td>
<td>FlipZ</td>
</tr>
<tr>
<td>HasExtensionDictionary</td>
<td>Handle</td>
</tr>
<tr>
<td>GetExtensionDictionary</td>
<td>Node</td>
</tr>
<tr>
<td>GetXData</td>
<td>Modified</td>
</tr>
<tr>
<td>SetXData</td>
<td>ObjectId</td>
</tr>
<tr>
<td>ObjectName</td>
<td>Offset</td>
</tr>
<tr>
<td>OwnerID</td>
<td>Reference</td>
</tr>
<tr>
<td>UseNodeCS</td>
<td>XRotation</td>
</tr>
<tr>
<td>YRotation</td>
<td>ZRotation</td>
</tr>
</tbody>
</table>
Attaches objects to volumes on a 3D layout grid.

**VBA object name:** AecAnchorEntToLayoutVolume

**Create using:** New AecAnchorEntToLayoutVolume

**Access via:** AecGeo.GetAnchor

<table>
<thead>
<tr>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
</tr>
<tr>
<td>ApplyResize</td>
</tr>
<tr>
<td>CenterOnNode</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Document</td>
</tr>
</tbody>
</table>
Methods

- AttachEntity
- Delete
- GetExtensionDictionary
- GetXData
- SetXData
- FlipX
- FlipY
- FlipZ
- Handle
- HasExtensionDictionary
- Node
- Modified
- Events

Attributes

- ObjectID
- ObjectName
- Offset
- OwnerID
- Reference
- ResizeOffset
- UseNodeCS
- XRotation
- YRotation
- ZRotation
**AecAnchorExtendedTagToEnt Object**

Attaches a tag to an object to display schedule data.

**VBA object name:** AecAnchorExtendedTagToEnt

**Create using:** New AecAnchorExtendedTagToEnt

**Access via:** AecGeo.GetAnchor

This object is available if the drawing contains any MVBlock styles.

---

**Properties**

- Application
- Description
- Document
- ForceHorizontal

**Methods**

- AttachEntity
- Delete
GetExtensionDictionary
GetXData
SetXData
Handle
HasExtensionDictionary
ObjectId
ObjectName
OwnerId
Reference
Connects an object to a layout node with a leader between the two objects.

**VBA object name:** AecAnchorLeadEntToNode

**Create using:** New AecAnchorLeadEntToNode

**Access via:** AecGeo.GetAnchor

<table>
<thead>
<tr>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>AngleFromNode</td>
</tr>
<tr>
<td>Application</td>
</tr>
<tr>
<td>CenterOnNode</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Document</td>
</tr>
</tbody>
</table>
Methods

**AttachEntity**

**Delete**

**GetXData**

**SetXData**

**Handle**

**HasExtensionDictionary**

**LeaderExtension1**

**LeaderExtension2**

**Modified**

**Node**

**ObjectId**

**ObjectName**

**Offset**

**OwnerID**

**Reference**

**UseNodeCS**

**XRotation**

**YRotation**

**ZRotation**
**Anchor to Reference**

Base class for referencing objects from an anchor.

**VBA object name:** AecAnchorToRef

**Create using:** New AecAnchorToRef

**Access via:** AecGeo.GetAnchor

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>AttachEntity</td>
</tr>
<tr>
<td>Description</td>
<td>Delete</td>
</tr>
<tr>
<td>Document</td>
<td>GetExtensionDictionary</td>
</tr>
<tr>
<td>Handle</td>
<td>HasExtensionDictionary</td>
</tr>
<tr>
<td>Modified</td>
<td>Events</td>
</tr>
<tr>
<td>GetXData</td>
<td>ObjectId</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>SetXData</td>
<td>ObjectName</td>
</tr>
<tr>
<td></td>
<td>OwnerID</td>
</tr>
<tr>
<td></td>
<td>Reference</td>
</tr>
</tbody>
</table>
Object

Application object extending AcadApplication to provide support for AEC Applications.

VBA object name: AecBaseApplication

Create using:

For VB:
GetObject("Acad.Application") or
CreateObject("Acad.Application") then
GetInterfaceObject("AecBase.Application")

For VBA:
not applicable. The application is always available.

Access via: Application Property

The properties associated with the AecBaseApplication 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 (Autodesk Architectural Desktop drawing) can be accessed using the `ActiveDocument` property. See the `AcadApplication` object in the AutoCAD ActiveX and VBA Reference for information regarding the Methods, Properties and Events provided through this object.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eval</td>
<td>ActiveDocument</td>
</tr>
<tr>
<td>GetAcadState</td>
<td>Application</td>
</tr>
<tr>
<td>GetInterfaceObject</td>
<td>Caption</td>
</tr>
<tr>
<td>Init</td>
<td>Documents</td>
</tr>
<tr>
<td>ListArx</td>
<td>FullName</td>
</tr>
<tr>
<td>LoadArx</td>
<td>Height</td>
</tr>
<tr>
<td>LoadDVB</td>
<td>LocaleId</td>
</tr>
<tr>
<td>Quit</td>
<td>MenuBar</td>
</tr>
<tr>
<td>RunMacro</td>
<td>MenuGroups</td>
</tr>
<tr>
<td>UnloadArx</td>
<td>Name</td>
</tr>
<tr>
<td>UnloadDVB</td>
<td>Path</td>
</tr>
<tr>
<td>Update</td>
<td>Preferences</td>
</tr>
<tr>
<td>ZoomAll</td>
<td>StatusId</td>
</tr>
<tr>
<td>ZoomCenter</td>
<td>VBE</td>
</tr>
<tr>
<td>ZoomExtents</td>
<td>Version</td>
</tr>
<tr>
<td>ZoomPickWindow</td>
<td>Visible</td>
</tr>
</tbody>
</table>
ZoomPrevious  Width
ZoomScaled     WindowLeft
ZoomWindow     WindowState
              WindowTop
AEC database object extending AcadDatabase to provide support for general AEC objects.

**VBA object name:** AecBaseDatabase

**Create using:** N/A

**Access via:** ModelSpace.Item

**Note:** All properties and methods of the AecBaseDatabase object are available through the AecBaseDocument object.

**Properties**

- Blocks
- Dictionaries
- DimStyles
Methods

CopyObjects
HandleToObject
Init
ObjectIdToObject

DisplayConfigurations
DisplayRepresentations
DisplaySets
ElevationModelSpace
ElevationPaperSpace
Groups
LayerKeyStyles
Layers

Layouts
Limits
Linetypes
MaskBlockStyles
MassElementStyles
MassGroups
ModelSpace
MVBlocKStyles
PaperSpace
PlotConfigurations
PolygonStyles
Preferences
ProfileStyles
RegisteredApplications
TextStyles
UserCoordinateSystems
Viewports
Views
Preferences Object

A derivative of AecDatabasePreferences, providing base desktop database preferences support.

VBA object name: AecBaseDatabasePreferences

Create using: N/A

Access via: AecBaseDatabase.Preferences
AecBaseDocument.Preferences

Properties

- AllowLongSymbolNames
- AlwaysImportLayerStandard
- AngularAzimuth
- AngularDisplayFormat
AngularPrecision
Application
AreaDisplayUnit
AreaPrecision
AreaSuffix
BasePoint
BasePointNE
BlockBasedLayerOffBehavior
ContourLinesPerSurface
CoordinatePrecision
CreateDimscaleOverride
Database
DatabaseScale
DisplaySilhouette
ElevationPrecision
FacetDeviation
FacetMaxiumum
LayerFile
LayerStandard
LinearDisplayFormat
LinearPrecision
ConvertToCurrentVolumeDisplay
Init
SaveAsDefault
LinearUnit
Lineweight
LineWeightDisplay
MaxActiveViewports
MeasurementUnit
NorthRotation
ObjectSortByPlotting
ObjectSortByPSOutput
ObjectSortByRedraws
ObjectSortByRegens
ObjectSortBySelection
ObjectSortBySnap
OLELaunch
ProjectName
RenderSmoothness
ScaleOnInsert
SegmentPerPolyline
SolidFill
TextFrameDisplay
TextHeight
VerticalScale
<table>
<thead>
<tr>
<th>Configuration Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertical Scale</td>
</tr>
<tr>
<td>VolumeDisplayUnit</td>
</tr>
<tr>
<td>VolumePrecision</td>
</tr>
<tr>
<td>VolumeSuffix</td>
</tr>
<tr>
<td>XRefEdit</td>
</tr>
<tr>
<td>XRefLayerVisibility</td>
</tr>
<tr>
<td>XrefOverlaysUseOwnDisplayConfig</td>
</tr>
</tbody>
</table>
An AEC drawing object, extending the AcadDocument to provide support for general AEC objects.

Note: This object is actually a component of the AecXUIBase object model, which represents user interface objects.

**VBA object name:** AecBaseDocument

**Create using:**
- Documents.Add
- Documents.Open
- Document.New

**Access via:**
- Documents.Item
- Application.ActiveDocument

The [Preferences](#) property provides access to drawing settings.

The active document can be accessed through the ActiveDocument property of the AecBaseApplication object, after initializing
AecBaseApplication. For example:

Dim doc As AecBaseDocument
Dim app As New AecBaseApplication

app.Init ThisDrawing.Application
Set doc = app.ActiveDocument

<table>
<thead>
<tr>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
</tr>
<tr>
<td>ActiveDimStyle</td>
</tr>
<tr>
<td>ActiveLayer</td>
</tr>
<tr>
<td>ActiveLayout</td>
</tr>
<tr>
<td>ActiveLinetype</td>
</tr>
<tr>
<td>ActivePViewport</td>
</tr>
<tr>
<td>ActiveSelectionSet</td>
</tr>
<tr>
<td>ActiveSpace</td>
</tr>
<tr>
<td>ActiveTextStyle</td>
</tr>
<tr>
<td>ActiveUCS</td>
</tr>
<tr>
<td>ActiveViewport</td>
</tr>
<tr>
<td>Application</td>
</tr>
<tr>
<td>Blocks</td>
</tr>
<tr>
<td>Database</td>
</tr>
<tr>
<td>Dictionaries</td>
</tr>
</tbody>
</table>
Methods

Activate
AuditInfo
Close
CopyObjects
EndUndoMark
Export
GetVariable
HandleToObject
Import
Init
LoadShapeFile
New
ObjectIdToObject
Open
PurgeAll
Regen
Save

DimStyles
DisplayConfigurations
DisplayRepresentations
DisplaySets
ElevationModelSpace
ElevationPaperSpace
FullName
Groups
Height
HWND
LayerKeyStyles
Layers
Layouts
Limits
Linetypes
MaskBlockStyles
MassElementStyles
MassGroups
ModelSpace
MSpace
<table>
<thead>
<tr>
<th>SaveAs</th>
<th>MVBlockStyles</th>
</tr>
</thead>
<tbody>
<tr>
<td>SendCommand</td>
<td>Name</td>
</tr>
<tr>
<td>SetVariable</td>
<td>ObjectSnapMode</td>
</tr>
<tr>
<td>StartUndoMark</td>
<td>PaperSpace</td>
</tr>
<tr>
<td>Wblock</td>
<td>Path</td>
</tr>
<tr>
<td></td>
<td>PickfirstSelectionSet</td>
</tr>
<tr>
<td></td>
<td>Plot</td>
</tr>
<tr>
<td></td>
<td>PlotConfigurations</td>
</tr>
<tr>
<td></td>
<td>PolygonStyles</td>
</tr>
<tr>
<td></td>
<td>Preferences</td>
</tr>
<tr>
<td></td>
<td>ProfileStyles</td>
</tr>
<tr>
<td></td>
<td>ReadOnly</td>
</tr>
<tr>
<td></td>
<td>RegisteredApplications</td>
</tr>
<tr>
<td></td>
<td>Saved</td>
</tr>
<tr>
<td></td>
<td>SelectionSets</td>
</tr>
<tr>
<td></td>
<td>TextStyles</td>
</tr>
<tr>
<td></td>
<td>UserCoordinateSystems</td>
</tr>
<tr>
<td></td>
<td>Utility</td>
</tr>
<tr>
<td></td>
<td>Viewports</td>
</tr>
<tr>
<td></td>
<td>Views</td>
</tr>
</tbody>
</table>
Width

WindowState

WindowTitle
Aec Base Documents Collection

The collection of all AEC drawings open in the current session.

Note: This object is actually a component of the AecXUIBase object model, which represents user interface objects.

VBA object name: AecBaseDocuments

Create using: N/A

Access via: AecBaseApplication.Documents

Methods

Add
Close
Init
Item

Properties

Application
Count
Item
Open
AEC AutoCAD preferences object.

**VBA object name:** AecBasePreferences

**Create using:** N/A

**Access via:** AecBaseApplication.Preferences

The AecBasePreferences object holds all the options stored external to the drawing, such as User Preferences. The options are stored in separate objects, grouped by category. Use the properties of the AecBasePreferences object to access each category.

Options that reside in the drawing can be accessed through the AecBaseDatabasePreferences object.

Currently, AecBasePreferences does not provide access to AEC-specific options; it is equivalent to AcadPreferences. Refer to AcadPreferences in the AutoCAD ActiveX and VBA Reference for information about this
object's methods and properties.

**Properties**
- Application
- Display
- Drafting
- Files

**Methods**
- Init
- OpenSave
- Output
- Profiles
- Selection
- System
- User
An instance of an AEC block in a drawing.

**VBA object name:** AecBlockRef

**Create using:** N/A

**Access via:**
- ModelSpace.Item
- PaperSpace.Item
- Block.Item
- SelectionSet.Item
- Group.Item

AecBlockRef is the base object for the AecMVBlockRef and AecMaskBlockRef objects.

**Properties**

<table>
<thead>
<tr>
<th>Methods</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td></td>
</tr>
<tr>
<td>ArrayPolar</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>ArrayRectangular</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>AttachAnchor</td>
<td>Document</td>
</tr>
<tr>
<td>Copy</td>
<td>Handle</td>
</tr>
<tr>
<td>Delete</td>
<td>HasExtensionDictionary</td>
</tr>
<tr>
<td>GetAnchor</td>
<td>Hyperlinks</td>
</tr>
<tr>
<td>GetBoundingBox</td>
<td>Layer</td>
</tr>
<tr>
<td>GetExtensionDictionary</td>
<td>Linetype</td>
</tr>
<tr>
<td>GetXData</td>
<td>LinetypeScale</td>
</tr>
<tr>
<td>Highlight</td>
<td>Lineweight</td>
</tr>
<tr>
<td>IntersectWith</td>
<td>Location</td>
</tr>
<tr>
<td>Mirror</td>
<td>Normal</td>
</tr>
<tr>
<td>Mirror3D</td>
<td>ObjectId</td>
</tr>
<tr>
<td>Move</td>
<td>ObjectName</td>
</tr>
<tr>
<td>ReleaseAnchor</td>
<td>OwnerID</td>
</tr>
<tr>
<td>Rotate</td>
<td>PlotStyleName</td>
</tr>
<tr>
<td>Rotate3D</td>
<td>Rotation</td>
</tr>
<tr>
<td>ScaleEntity</td>
<td>ScaleX</td>
</tr>
<tr>
<td>SetXData</td>
<td>ScaleY</td>
</tr>
<tr>
<td>TransformBy</td>
<td>ScaleZ</td>
</tr>
<tr>
<td>Update</td>
<td>TrueColor</td>
</tr>
<tr>
<td></td>
<td>Visible</td>
</tr>
</tbody>
</table>
AEC camera object.

**VBA object name:** AecCamera

**Create using:** AddCustomObject("AecCamera")

**Access via:**
- ModelSpace.Item
- PaperSpace.Item
- Block.Item
- SelectionSet.Item
- Group.Item

**Methods**
- `ArrayPolar`
- `ArrayRectangular`
- `AttachAnchor`

**Properties**
- `Application`
- `Color`
- `Description`
- `Document`
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy</td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td></td>
</tr>
<tr>
<td>GetAnchor</td>
<td></td>
</tr>
<tr>
<td>GetBoundingBox</td>
<td></td>
</tr>
<tr>
<td>GetExtensionDictionary</td>
<td></td>
</tr>
<tr>
<td>GetXData</td>
<td></td>
</tr>
<tr>
<td>Highlight</td>
<td></td>
</tr>
<tr>
<td>IntersectWith</td>
<td></td>
</tr>
<tr>
<td>Mirror</td>
<td></td>
</tr>
<tr>
<td>Mirror3D</td>
<td></td>
</tr>
<tr>
<td>Move</td>
<td></td>
</tr>
<tr>
<td>ReleaseAnchor</td>
<td></td>
</tr>
<tr>
<td>Rotate</td>
<td></td>
</tr>
<tr>
<td>Rotate3D</td>
<td></td>
</tr>
<tr>
<td>ScaleEntity</td>
<td></td>
</tr>
<tr>
<td>SetXData</td>
<td></td>
</tr>
<tr>
<td>TransformBy</td>
<td></td>
</tr>
<tr>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>Handle</td>
<td></td>
</tr>
<tr>
<td>HasExtensionDictionary</td>
<td></td>
</tr>
<tr>
<td>Hyperlinks</td>
<td></td>
</tr>
<tr>
<td>Layer</td>
<td></td>
</tr>
<tr>
<td>Linetype</td>
<td></td>
</tr>
<tr>
<td>LinetypeScale</td>
<td></td>
</tr>
<tr>
<td>Lineweight</td>
<td></td>
</tr>
<tr>
<td>Location</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>ObjectID</td>
<td></td>
</tr>
<tr>
<td>ObjectName</td>
<td></td>
</tr>
<tr>
<td>OwnerID</td>
<td></td>
</tr>
<tr>
<td>PlotStyleName</td>
<td></td>
</tr>
<tr>
<td>Rotation</td>
<td></td>
</tr>
<tr>
<td>TrueColor</td>
<td></td>
</tr>
<tr>
<td>Visible</td>
<td></td>
</tr>
<tr>
<td>Zoom</td>
<td></td>
</tr>
</tbody>
</table>
Cell Layout Tool

**Object**

Base Class for the layout grid objects.

**VBA object name:** AecCellLayoutTool

**Create using:** N/A

**Access via:**
- ModelSpace.Item
- PaperSpace.Item
- Block.Item
- SelectionSet.Item
- Group.Item

This is the base class for all rectangular cell layout tools, including AecLayoutGrid2D and AecLayoutGrid3d. You can use AecCellLayoutTool as a generic reference to layout grid objects.

**Methods**

- **ArrayPolar**
<table>
<thead>
<tr>
<th>Command</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrayRectangular</td>
<td>Application</td>
</tr>
<tr>
<td>AttachAnchor</td>
<td>Color</td>
</tr>
<tr>
<td>ClosestNode</td>
<td>Description</td>
</tr>
<tr>
<td>Copy</td>
<td>Document</td>
</tr>
<tr>
<td>Delete</td>
<td>Handle</td>
</tr>
<tr>
<td>GetAnchor</td>
<td>HasExtensionDictionary</td>
</tr>
<tr>
<td>GetBoundingBox</td>
<td>Hyperlinks</td>
</tr>
<tr>
<td>GetExtensionDictionary</td>
<td>Hyperlinks</td>
</tr>
<tr>
<td>GetXData</td>
<td>Layer</td>
</tr>
<tr>
<td>Highlight</td>
<td>Linetype</td>
</tr>
<tr>
<td>IntersectWith</td>
<td>LinetypeScale</td>
</tr>
<tr>
<td>Mirror</td>
<td>Lineweight</td>
</tr>
<tr>
<td>Mirror3D</td>
<td>Location</td>
</tr>
<tr>
<td>Move</td>
<td>Normal</td>
</tr>
<tr>
<td>NodeLocation</td>
<td>ObjectId</td>
</tr>
<tr>
<td>ReleaseAnchor</td>
<td>ObjectName</td>
</tr>
<tr>
<td>Rotate</td>
<td>OwnerID</td>
</tr>
<tr>
<td>Rotate3D</td>
<td>PlotStyleName</td>
</tr>
<tr>
<td>ScaleEntity</td>
<td>Rotation</td>
</tr>
<tr>
<td>SetXData</td>
<td>TrueColor</td>
</tr>
<tr>
<td>TransformBy</td>
<td>Visible</td>
</tr>
</tbody>
</table>
Update
Clip Volume entity defining an extruded region for clipping.

**VBA object name:** AecClipVol

**Create using:** N/A

**Access via:**
- ModelSpace.Item
- PaperSpace.Item
- Block.Item
- SelectionSet.Item
- Group.Item

This is the base class for ADT building elevation and section line objects.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle1</td>
<td>ArrayPolar</td>
</tr>
<tr>
<td>Angle2</td>
<td>Application</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>ArrayRectangular</td>
<td>Color</td>
</tr>
<tr>
<td>AttachAnchor</td>
<td>Description</td>
</tr>
<tr>
<td>ClosestNode</td>
<td>Document</td>
</tr>
<tr>
<td>Copy</td>
<td>Handle</td>
</tr>
<tr>
<td>Delete</td>
<td>HasExtensionDictionary</td>
</tr>
<tr>
<td>GetAnchor</td>
<td>Height</td>
</tr>
<tr>
<td>GetBoundingBox</td>
<td>Hyperlinks</td>
</tr>
<tr>
<td>GetExtensionDictionary</td>
<td>Layer</td>
</tr>
<tr>
<td>GetXData</td>
<td>Linetype</td>
</tr>
<tr>
<td>Highlight</td>
<td>LinetypeScale</td>
</tr>
<tr>
<td>IntersectWith</td>
<td>Lineweight</td>
</tr>
<tr>
<td>Mirror</td>
<td>Location</td>
</tr>
<tr>
<td>Mirror3D</td>
<td>LowerExtension</td>
</tr>
<tr>
<td>Move</td>
<td>Normal</td>
</tr>
<tr>
<td>NodeLocation</td>
<td>ObjectID</td>
</tr>
<tr>
<td>ReleaseAnchor</td>
<td>ObjectName</td>
</tr>
<tr>
<td>Rotate</td>
<td>OwnerID</td>
</tr>
<tr>
<td>Rotate3D</td>
<td>PlotStyleName</td>
</tr>
<tr>
<td>ScaleEntity</td>
<td>Rotation</td>
</tr>
<tr>
<td>SetXData</td>
<td>Side1</td>
</tr>
<tr>
<td>TransformBy</td>
<td>Side2</td>
</tr>
<tr>
<td>------------</td>
<td>-------</td>
</tr>
<tr>
<td>Update</td>
<td>TrueColor</td>
</tr>
<tr>
<td>UseModelExtents</td>
<td>Visible</td>
</tr>
</tbody>
</table>
Clip Volume Result which captures the resulting graphics for a section or elevation for insertion into the drawing.

VBA object name: AecClipVolRes

Create using: N/A

Access via:
- ModelSpace.Item
- PaperSpace.Item
- Block.Item
- SelectionSet.Item
- Group.Item

This is the base class for the ADT section line object.

Methods
- ArrayPolar
- ArrayRectangular

Properties
- Application
<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttachAnchor</td>
<td>Color</td>
</tr>
<tr>
<td>Copy</td>
<td>Description</td>
</tr>
<tr>
<td>Delete</td>
<td>Document</td>
</tr>
<tr>
<td>GetAnchor</td>
<td>Handle</td>
</tr>
<tr>
<td>GetBoundingBox</td>
<td>HasExtensionDictionary</td>
</tr>
<tr>
<td>GetExtensionDictionary</td>
<td>Hyperlinks</td>
</tr>
<tr>
<td>GetXData</td>
<td>Layer</td>
</tr>
<tr>
<td>Highlight</td>
<td>Linetype</td>
</tr>
<tr>
<td>IntersectWith</td>
<td>LinetypeScale</td>
</tr>
<tr>
<td>Mirror</td>
<td>Lineweight</td>
</tr>
<tr>
<td>Mirror3D</td>
<td>Location</td>
</tr>
<tr>
<td>Move</td>
<td>Normal</td>
</tr>
<tr>
<td>ReleaseAnchor</td>
<td>ObjectID</td>
</tr>
<tr>
<td>Rotate</td>
<td>ObjectName</td>
</tr>
<tr>
<td>Rotate3D</td>
<td>OwnerID</td>
</tr>
<tr>
<td>ScaleEntity</td>
<td>PlotStyleName</td>
</tr>
<tr>
<td>SetXData</td>
<td>Rotation</td>
</tr>
<tr>
<td>TransformBy</td>
<td>TrueColor</td>
</tr>
<tr>
<td>Update</td>
<td>Visible</td>
</tr>
</tbody>
</table>
AEC Database object extending the Acad Database object.

**VBA object name:** AecDatabase

**Create using:** N/A

**Access via:** ModelSpace.Item

Note: Access all properties and methods of the AecDatabase object through the [AecBaseDocument](#) object.

<table>
<thead>
<tr>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blocks</td>
</tr>
<tr>
<td>Dictionaries</td>
</tr>
<tr>
<td>DimStyles</td>
</tr>
<tr>
<td>ElevationModelSpace</td>
</tr>
<tr>
<td>ElevationPaperSpace</td>
</tr>
</tbody>
</table>
Methods

Elevation
PaperSpace

Groups

Layers

HandleToObject

Init

ObjectIdToObject

ObjectIdsToObject

PaperSpace

PlotConfigurations

Preferences

RegisteredApplications

TextStyles

UserCoordinateSystems

Viewports

Views
Preferences Object

Extends the AutoCAD Database Preferences object to include AEC properties and methods.

VBA object name: AecDatabasePreferences

Create using: N/A

Access via: ModelSpace.Item

Note: Use AecBaseDatabasePreferences instead of AecDatabasePreferences.

Properties

- AllowLongSymbolNames
- Application
- ContourLinesPerSurface
Methods

Init

DisplaySilhouette
Lineweight
LineWeightDisplay
MaxActiveViewports

ObjectSortByPlotting
ObjectSortByPSOutput
ObjectSortByRedraws
ObjectSortByRegens
ObjectSortByRegens
ObjectSortBySelection
ObjectSortBySnap
OLELaunch
RenderSmoothness
SegmentPerPolyline
SolidFill
TextFrameDisplay
XRefEdit
XRefLayerVisibility
AEC Dictionary containing a collection of AEC objects.

**VBA object name:** AecDictionary

**Create using:** N/A

**Access via:** N/A

AecDictionary is the base class for AEC style collections, such as AecLayerKeyStyles and AecMaskBlockStyles. These collections can be accessed from the AecBaseDocument object.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete</td>
<td>Application</td>
<td></td>
</tr>
<tr>
<td>GetExtensionDictionary</td>
<td>Count</td>
<td></td>
</tr>
<tr>
<td>GetXData</td>
<td>Document</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Handle</td>
<td></td>
</tr>
<tr>
<td>Has</td>
<td>HasExtensionDictionary</td>
<td>Modified</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Remove</td>
<td>ObjectID</td>
<td></td>
</tr>
<tr>
<td>Rename</td>
<td>ObjectName</td>
<td></td>
</tr>
<tr>
<td>SetXData</td>
<td>OwnerID</td>
<td></td>
</tr>
</tbody>
</table>
An AEC Dictionary item.

**VBA object name:** AecDictRecord

**Create using:** N/A

**Access via:** N/A

This is the base class for all AEC Style objects, such as AecLayerKeyStyle and AecMaskBlockStyle.

### Properties

- **AlternateName**
- **Application**

### Methods

- **Delete**
- **Description**
- **Document**
<table>
<thead>
<tr>
<th>Method</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetExtensionDictionary</td>
<td>Handle</td>
</tr>
<tr>
<td>GetXData</td>
<td>HasExtensionDictionary</td>
</tr>
<tr>
<td>SetXData</td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td>ObjectID</td>
</tr>
<tr>
<td></td>
<td>ObjectName</td>
</tr>
<tr>
<td></td>
<td>OwnerID</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modified</td>
</tr>
</tbody>
</table>
AecDisplayComponent

Object

An entity or hatch component of a the AecDisplayComponents collection.

VBA object name: AecDisplayComponent

Create using: N/A

Access via: AecDisplayComponents.Item

AecDisplayComponent is a base class. The derived classes are AecDisplayComponentEntity or AecDisplayComponentHatch. Check the object type returned by the AecDisplayComponents.Item for the underlying object.
Owner
AecDisplayComponentEntity Object

The entity properties for the display component.

VBA object name: AecDisplayComponentEntity

Create using: N/A

Access via: AecDisplayComponents.Item

AecDisplayComponentEntity is a derived class. AecDisplayComponent is the base class. Check the type returned by the AecDisplayComponents.Item method for the underlying object.

Properties

AllowByMaterial

ByMaterial

Index
<table>
<thead>
<tr>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer</td>
<td></td>
</tr>
<tr>
<td>Linetype</td>
<td></td>
</tr>
<tr>
<td>Linetypescale</td>
<td></td>
</tr>
<tr>
<td>LineWeight</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>Owner</td>
<td></td>
</tr>
<tr>
<td>PlotStyleName</td>
<td></td>
</tr>
<tr>
<td>TrueColor</td>
<td></td>
</tr>
<tr>
<td>Visible</td>
<td></td>
</tr>
</tbody>
</table>
The hatch properties for the display component.

**VBA object name:** AecDisplayComponentHatch

**Create using:** N/A

**Access via:** AecDisplayComponents.Item

AecDisplayComponentHatch is a derived class. AecDisplayComponent is the base class. Check the type returned by the AecDisplayComponents.Item method for the underlying object.

**Properties**

- **Angle**
- **DoubleHatch**
- **HatchType**
Methods

Index

Name

Owner

PatternName

Scale

Spacing

UseAngleOfObject

XOffset

YOffset

Events
AecDisplayComponents Collection

The collection of display components in a display representation.

**VBA object name:** AecDisplayComponents

**Create using:** N/A

**Access via:** AecDisplayProperties.DisplayComponents

The display components collection contains AecDisplayComponentEntity and AecDisplayComponent Hatch objects.

**Properties**

- Count
  - Item
  - Owner

**Methods**

- Item
  - Properties
  - Events
AecDisplayConfiguration Object

A display configuration.

**VBA object name:** AecDisplayConfiguration

**Create using:** N/A

**Access via:** AecDisplayConfiguration.Item

### Properties

- **AlternateName**
- **CutPlaneDisplayAbove**
- **CutPlaneDisplayBelow**
- **CutPlaneHeight**

### Methods

- **DisplaySet**

### Events

- **Modified**
FixedViewDirection

Name

UseFixedViewDirection
AecDisplayConfigurations Object

A collection of display configurations.

**VBA object name:** AecDisplayConfigurations

**Create using:** N/A

**Access via:**
- AecBaseDatabase.DisplayConfigurations
- AecBaseDocument.DisplayConfigurations

**Methods**
- Add
- Has
- Import
- Item
- Remove

**Properties**
- ActiveConfiguration
- Count

**Events**
- Modified
- Modified
Rename
Object

A collection of display properties for a display representation.

**VBA object name**: AecDisplayProperties

**Create using**: N/A

**Access via**: AecDisplayRepresentation.DefaultDisplayProperties
AecDisplayRepresentation.OverrideDisplayProperties

Use the AecDisplayProperties collection to access the material properties, entity component properties and hatch component properties of the display representation.

AecDisplayProperties is a base class. The derived class is AecDisplayPropertiesMaterial. Check the type returned by the DefaultDisplayProperties method and OverrideDisplayProperties method for the underlying object.
Methods

DisplayComponents

Events
AecDisplayPropertiesMaterial Object

The material display properties for a display representation.

**VBA object name:** AecDisplayPropertiesMaterial

**Create using:** N/A

**Access via:**
- AecDisplayRepresentation.DefaultDisplayProperties
- AecDisplayRepresentation.OverrideDisplayProperties

AecDisplayPropertiesMaterial is a derived class. AecDisplayProperties is the base class. Check the type returned by the DefaultDisplayProperties method and OverrideDisplayProperties method for the underlying object.

**Properties**

- DisplayComponents
- DisplayHiddenLinework
Methods

ExcludeFrom2dSectionShrinkwrap
SectionedBodyRenderingMaterialName
SectionRenderingMaterialName
SurfaceHatchPlacement
SurfaceRenderingMaterialName
SurfaceRenderMaterialMapping
MergeCommonMaterials

Events
**AecDisplayRepresentation Object**

A display representation.

**VBA object name:** AecDisplayRepresentation

**Create using:** AecDisplayRepresentation.Duplicate

**Access via:** AecDisplayRepresentations.Item

**Name**

<table>
<thead>
<tr>
<th>Properties</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AlternnameName</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AddOverride</td>
<td></td>
</tr>
<tr>
<td>ClassName</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Events</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultDisplayProperties</td>
<td></td>
</tr>
<tr>
<td>Modified</td>
<td></td>
</tr>
</tbody>
</table>
OverrideDisplayProperties
# AecDisplayRepresentations Object

The collection of display representations.

**VBA object name:** AecDisplayRepresentations

**Create using:** N/A

**Access via:**
- AecDisplaySet.DisplayRepresentations
- AecBaseDatabase.DisplayRepresentations
- AecBaseDocument.DisplayRepresentations

## Methods
- Append
- Has
- Item
- Remove

## Properties
- Count
- Owner
- Events
**Object**

In-place edit profile object.

**VBA object name:** AecEditInPlaceProfile

**Create using:**

```
AddCustomObject("AecEditInPlaceProfile")
```

**Access via:**

- ModelSpace.Item
- PaperSpace.Item
- Block.Item
- SelectionSet.Item
- Group.Item

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrayPolar</td>
<td>Application</td>
</tr>
<tr>
<td>ArrayRectangular</td>
<td>Description</td>
</tr>
<tr>
<td>AttachAnchor</td>
<td>Document</td>
</tr>
</tbody>
</table>
Base class for all AEC entities.

**VBA object name:** AecEntity

**Create using:** N/A

**Access via:**
- ModelSpace.Item
- PaperSpace.Item
- Block.Item
- SelectionSet.Item
- Group.Item

Although AecEntity can be used as a generic reference to any AEC object, it is better to use AecGeo as a generic AEC object reference. AecGeo is the greatest common dominator for all AEC objects.

**Methods**
- ArrayPolar
- ArrayRectangular

**Properties**
- Application
A reference to any entity in the drawing, similar to a block reference.

**VBA object name:** AecEntRef

**Create using:** N/A

**Access via:**
- ModelSpace.Item
- PaperSpace.Item
- Block.Item
- SelectionSet.Item
- Group.Item

**Properties**
- Application

**Methods**
- Color
- ArrayPolar
- Description
UseOffset

Visible
AecGeo Object

Provides location, rotation and anchoring capabilities for AEC entities.

VBA object name: AecGeo

Create using: N/A

Access via:
- ModelSpace.Item
- PaperSpace.Item
- Block.Item
- SelectionSet.Item
- Group.Item

Use AecGeo as a generic reference to AEC objects.

Methods
- ArrayPolar
- ArrayRectangular
- AttachAnchor

Properties
- Application
- Color
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td>Document</td>
<td></td>
</tr>
<tr>
<td>GetAnchor</td>
<td>Handle</td>
<td></td>
</tr>
<tr>
<td>GetBoundingBox</td>
<td>HasExtensionDictionary</td>
<td></td>
</tr>
<tr>
<td>GetExtensionDictionary</td>
<td>Layer</td>
<td></td>
</tr>
<tr>
<td>GetXData</td>
<td>Linetype</td>
<td></td>
</tr>
<tr>
<td>Highlight</td>
<td>LinetypeScale</td>
<td>Events</td>
</tr>
<tr>
<td>IntersectWith</td>
<td>Lineweight</td>
<td>Modified</td>
</tr>
<tr>
<td>Mirror</td>
<td>Location</td>
<td></td>
</tr>
<tr>
<td>Mirror3D</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>Move</td>
<td>ObjectID</td>
<td></td>
</tr>
<tr>
<td>ReleaseAnchor</td>
<td>ObjectName</td>
<td></td>
</tr>
<tr>
<td>Rotate</td>
<td>OwnerID</td>
<td></td>
</tr>
<tr>
<td>Rotate3D</td>
<td>PlotStyleName</td>
<td></td>
</tr>
<tr>
<td>ScaleEntity</td>
<td>Rotation</td>
<td></td>
</tr>
<tr>
<td>SetXData</td>
<td>TrueColor</td>
<td></td>
</tr>
<tr>
<td>TransformBy</td>
<td>Visible</td>
<td></td>
</tr>
<tr>
<td>Update</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Base class for grid assemblies.

**VBA object name:** AecGridAssembly

**Create using:** N/A

**Access via:**
- ModelSpace.Item
- PaperSpace.Item
- Block.Item
- SelectionSet.Item
- Group.Item

You can use AecGridAssembly as a generic reference to grids, but no properties or methods specific to grids are currently exposed through this object.

**Methods**

**ArrayPolar**
<table>
<thead>
<tr>
<th>ArrayRectangular</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttachAnchor</td>
<td>Application</td>
</tr>
<tr>
<td>ClosestNode</td>
<td>Color</td>
</tr>
<tr>
<td>Copy</td>
<td>Description</td>
</tr>
<tr>
<td>Delete</td>
<td>Document</td>
</tr>
<tr>
<td>GetAnchor</td>
<td>HasExtensionDictionary</td>
</tr>
<tr>
<td>GetBoundingBox</td>
<td>Hyperlinks</td>
</tr>
<tr>
<td>GetExtensionDictionary</td>
<td>Layer</td>
</tr>
<tr>
<td>GetXData</td>
<td>Linetype</td>
</tr>
<tr>
<td>Highlight</td>
<td>LinetypeScale</td>
</tr>
<tr>
<td>IntersectWith</td>
<td>Lineweight</td>
</tr>
<tr>
<td>Mirror</td>
<td>Location</td>
</tr>
<tr>
<td>Mirror3D</td>
<td>Normal</td>
</tr>
<tr>
<td>Move</td>
<td>ObjectId</td>
</tr>
<tr>
<td>NodeLocation</td>
<td>ObjectName</td>
</tr>
<tr>
<td>ReleaseAnchor</td>
<td>OwnerID</td>
</tr>
<tr>
<td>Rotate</td>
<td>PlotStyleName</td>
</tr>
<tr>
<td>Rotate3D</td>
<td>Rotation</td>
</tr>
<tr>
<td>ScaleEntity</td>
<td>TrueColor</td>
</tr>
<tr>
<td>SetXData</td>
<td>Visible</td>
</tr>
<tr>
<td></td>
<td>Events</td>
</tr>
<tr>
<td></td>
<td>Modified</td>
</tr>
</tbody>
</table>
TransformBy

Update
Layer key definition, which specifies the layer properties generated from a particular key.

**VBA object name:** AecLayerKey

**Create using:** AecLayerKeys.Add

**Access via:** AecLayerKeys.Item

<table>
<thead>
<tr>
<th>Properties</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>Index</td>
<td></td>
</tr>
<tr>
<td>Layer</td>
<td></td>
</tr>
<tr>
<td>LayerKeyStyle</td>
<td></td>
</tr>
</tbody>
</table>

**Methods**
None

Linetype

LineWeight

Name

PlotStyleName

Plottable

Removeable
Object

The collection of layer key definitions in a layer key style.

VBA object name: AecLayerKeys

Create using: N/A

Access via: AecLayerKeyStyle.Keys

Methods

Add
Item
Remove

Properties

Count
LayerKeyStyle
A collection of layer keys.

**VBA object name:** AecLayerKeyStyle

**Create using:** N/A

**Access via:** AecLayerKeyStyles.Item

### Properties
- AlternateName
- Application
- Description
- Document
- Handle

### Methods
- Delete
- ExpandKey
ExpandKey

GenerateLayer

GetExtensionDictionary

GetXData

SetXData

HasExtensionDictionary

Keys

Name

ObjectId

ObjectName

OverridesEnabled

OverrideSettings

OwnerID

Events

Modified
Layer Key Styles

Collection

The collection of layer key styles in the specified drawing.

VBA object name: AecLayerKeyStyles

Create using: N/A

Access via: AecBaseDatabase.LayerKeyStyles

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Application</td>
</tr>
<tr>
<td>Delete</td>
<td>Count</td>
</tr>
<tr>
<td>GetExtensionDictionary</td>
<td>Document</td>
</tr>
<tr>
<td>GetXData</td>
<td>Handle</td>
</tr>
<tr>
<td>Has</td>
<td>HasExtensionDictionary</td>
</tr>
</tbody>
</table>

Events

Modified
<table>
<thead>
<tr>
<th>Item</th>
<th>ObjectID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove</td>
<td>ObjectName</td>
</tr>
<tr>
<td>SetXData</td>
<td>OwnerID</td>
</tr>
</tbody>
</table>
A layer key style override setting.

**VBA object name:** AecLayerOverrideSetting

**Create using:** N/A

**Access via:** AecLayerOverrideSettings.Item

**Properties**

- **Index**

**Methods**

- **LayerKeyStyle**
- None

- **Name**
- **Value**
**Layer Key Style**

### Override Settings Collection

The collection of override settings in a layer key style.

**VBA object name:** AecLayerOverrideSettings

**Create using:** N/A

**Access via:** AecLayerKeyStyle.OverrideSettings

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Count</td>
<td>LayerKeyStyle</td>
</tr>
</tbody>
</table>
A single dimensional grid with nodes along a single curve.

**VBA object name:** AecLayoutCurve

**Create using:** N/A

**Access via:**
- ModelSpace.Item
- PaperSpace.Item
- Block.Item
- SelectionSet.Item
- Group.Item

You cannot create a new layout curve through ActiveX, and many properties of this object are not yet exposed.
ArrayPolar  Count
ArrayRectangular  Description
AttachAnchor  Document
ClosestNode  EndOffset
Copy  Handle
Delete  HasExtensionDictionary
GetAnchor  Hyperlinks
GetBoundingBox  Layer
GetExtensionDictionary  Linetype
GetXData  LinetypeScale
Highlight  Lineweight
IntersectWith  Location
Mirror  Nodes
Mirror3D  Normal
Move  ObjectID
NodeLocation  ObjectName
ReleaseAnchor  OwnerID
Rotate  PlotStyleName
Rotate3D  Rotation
ScaleEntity  Spacing
SetXData  StartOffset
TransformBy  TrueColor
Update  Type
Value
Visible
A node in a layout curve.

**VBA object name:** AecLayoutCurveNode

**Create using:** N/A

**Access via:** AecLayoutCurveNodes.Item

Most node properties have not been exposed through ActiveX.

### Properties

- **Index**
- **LayoutCurve**
- **Spacing**
**Collection**

The collection of nodes in a layout curve.

**VBA object name:** AecLayoutCurveNodes

**Create using:** N/A

**Access via:** AecLayoutCurve.Nodes

You can access the members of the collection, but they currently contain no useful information.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Count</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Item</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LayoutCurve</td>
<td></td>
</tr>
</tbody>
</table>
Layout grid entity used to layout any AEC object within a 2D grid.

**VBA object name:** AecLayoutGrid2D

**Create using:**

```
AddCustomObject("AecLayoutGrid2D")
```

**Access via:**

- ModelSpace.Item
- PaperSpace.Item
- Block.Item
- SelectionSet.Item
- Group.Item

### Properties

- **Angle**
- **Application**
- **BayAngle**
Methods

ArrayPolar
ArrayRectangular
AttachAnchor
Copy
ClosestNode
Delete
GetAnchor
GetBoundingBox
GetExtensionDictionary
GetXData
Highlight
IntersectWith
Mirror
Mirror3D
Move

Boundary
Color
Depth
Description

Document

Handle
HasExtensionDictionary
Hyperlinks
InsideRadius
Layer
Linetype
LinetypeScale
Lineweight
Location
Normal

ObjectID
ObjectName
OwnerID
PlotStyleName
Rotation

Events
Modified
Move
NodeLocation
ReleaseAnchor
Rotate
Rotate3D
ScaleEntity
SetXData
TransformBy
Update
Shape
TrueColor
Visible
Width
XCount
XEndOffset
XNodes
XSpacing
XStartOffset
XType
YCount
YEndOffset
YNodes
YSpacing
YStartOffset
YType
ZCount
ZSpacing
Object

A node in a 2D layout grid.

**VBA object name:** AecLayoutGrid2DNode

**Create using:** N/A

**Access via:** AecLayoutGrid2DNodes.Item

(But see note below.)

This object is not currently accessible. You can access 2D node objects through AecLayoutGrid2DXNode and AecLayoutGrid2DYNode, but most of their properties are not exposed.

<table>
<thead>
<tr>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

Spacing
### Layout Grid 2D Nodes

#### Collection

The collection of nodes in a 2D layout grid.

**VBA object name:** AecLayoutGrid2DNodes

**Create using:** N/A

**Access via:** N/A

This collection is not accessible. Use [AecLayoutGrid2DXNodes](#) and [AecLayoutGrid2DYNodes](#) instead.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Count</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Item</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LayoutGrid2D</strong></td>
<td></td>
<td>None</td>
</tr>
</tbody>
</table>
### Node Object

A node along the X axis of a 2D layout grid.

**VBA object name:** AecLayoutGrid2DXNode

**Create using:** N/A

**Access via:** AecLayoutGrid2DXNodes.Item

Most node properties have not been exposed through ActiveX.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index</td>
<td>LayoutGrid2D</td>
<td>None</td>
</tr>
<tr>
<td>Spacing</td>
<td></td>
<td>None</td>
</tr>
</tbody>
</table>
Nodes Collection

Collection of nodes along the X axis of a 2D layout grid.

**VBA object name:** AecLayoutGrid2DXNodes

**Create using:** N/A

**Access via:** AecLayoutGrid2D.XNodes

You can access the members of this collection, but they contain no useful information.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Count</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
<td>Item</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>LayoutGrid2D</td>
<td>None</td>
</tr>
</tbody>
</table>
## Node Object

A node along the Y axis of a 2D layout grid.

**VBA object name:** AecLayoutGrid2DYNode

**Create using:** N/A

**Access via:** AecLayoutGrid2DYNodes.Item

Most node properties have not been exposed through ActiveX.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>LayoutGrid2D</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

Spacing
Nodes Collection

Collection of nodes along the Y axis of a 2D layout grid.

**VBA object name:** AecLayoutGrid2DYNodes

**Create using:** N/A

**Access via:** AecLayoutGrid2D.YNodes

You can access the members of this collection, but they contain no useful information.

**Properties**

- Count
- LayoutGrid2D

**Methods**

- Item

**Events**

- None
A layout volume for arranging objects in a 3D grid.

**VBA object name:** AecLayoutGrid3D

**Create using:** `AddCustomObject("AecLayoutGrid3D")`

**Access via:**
- ModelSpace.Item
- PaperSpace.Item
- Block.Item
- SelectionSet.Item
- Group.Item

**Properties**
- Application
- Color
- Depth
<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
<th>Document</th>
<th>Handle</th>
<th>HasExtensionDictionary</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrayPolar</td>
<td>Hyperlinks</td>
<td>Layer</td>
<td>Linetype</td>
<td>LinetypeScale</td>
<td>Location</td>
</tr>
<tr>
<td>ArrayRectangular</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Normal</td>
</tr>
<tr>
<td>AttachAnchor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ObjectID</td>
</tr>
<tr>
<td>ClosestNode</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ObjectName</td>
</tr>
<tr>
<td>Copy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>OwnerID</td>
</tr>
<tr>
<td>Delete</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PlotStyleName</td>
</tr>
<tr>
<td>GetAnchor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Rotation</td>
</tr>
<tr>
<td>GetBoundingBox</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TrueColor</td>
</tr>
<tr>
<td>GetExtensionDictionary</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Visible</td>
</tr>
<tr>
<td>GetXData</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Width</td>
</tr>
<tr>
<td>Highlight</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IntersectWith</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mirror</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Events</td>
</tr>
<tr>
<td>Mirror3D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Modified</td>
</tr>
<tr>
<td>Move</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
move
NodeLocation
ReleaseAnchor
Rotate
Rotate3D
ScaleEntity
SetXData
TransformBy
Update
XCount
XEndOffset
XNodes
XSpacing
XStartOffset
XType
YCount
YEndOffset
YNodes
YSpacing
YStartOffset
YType
ZCount
ZEndOffset
ZNodes
ZSpacing
ZStartOffset
ZType
Object

A node of a 3D Layout Grid.

**VBA object name:** AecLayoutGrid3DNode

**Create using:** N/A

**Access via:** AecLayoutGrid3DNodes.Item

(But see note below.)

This object is not currently accessible. You can access 3D node objects through [AecLayoutGrid3DXNode](#) and [AecLayoutGrid3DYNode](#), but most of their properties are not exposed.

### Properties

- **Index**
- **Spacing**

### Methods

None

### Events

- **LayoutGrid3D** None
The collection of nodes in a 3D layout grid.

**VBA object name:** AecLayoutGrid3DNodes

**Create using:** N/A

**Access via:** N/A

This collection is not accessible. Use AecLayoutGrid3DXNodes and AecLayoutGrid3DYNodes instead.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Count</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td></td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>LayoutGrid3D</td>
<td></td>
</tr>
</tbody>
</table>
Node Object

A node along the X axis of a layout volume.

VBA object name: AecLayoutGrid3DXNode

Create using: N/A

Access via: AecLayoutGrid3DXNodes.Item

Most node properties have not been exposed through ActiveX.

Properties

Index

Methods

None

LayoutGrid3D

None

Events

Spacing
Nodes Collection

Collection of nodes along the X axis of a layout volume.

**VBA object name:** AecLayoutGrid3DXNodes

**Create using:** N/A

**Access via:** AecLayoutGrid3D.XNodes

You can access the members of this collection, but they contain no useful information.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Count</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Item</strong></td>
</tr>
<tr>
<td></td>
<td><strong>LayoutGrid3D</strong></td>
</tr>
</tbody>
</table>
Node Object

A node along the Y axis of a layout volume.

**VBA object name:** AecLayoutGrid3DYNode

**Create using:** N/A

**Access via:** AecLayoutGrid3DYNodes.Item

Most node properties have not been exposed through ActiveX.

**Properties**

<table>
<thead>
<tr>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Spacing</strong></td>
</tr>
</tbody>
</table>

**Methods**

None

**Events**

None

**LayoutGrid3D**

None

**Spacing**
Nodes Collection

Collection of nodes along the Y axis of a layout volume.

**VBA object name:** AecLayoutGrid3DYNodes

**Create using:** N/A

**Access via:** AecLayoutGrid3D.YNodes

You can access the members of this collection, but they contain no useful information.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Count</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LayoutGrid3D</td>
</tr>
</tbody>
</table>
Node Object

A node along the Z axis of a layout volume.

**VBA object name:** AecLayoutGrid3DZNode

**Create using:** N/A

**Access via:** AecLayoutGrid3DZNodes.Item

Most node properties have not been exposed through ActiveX.

**Properties**

- **Index**

**Methods**

- None
  - **LayoutGrid3D**
  - **Spacing**
Nodes Collection

Collection of nodes along the Z axis of a layout volume.

**VBA object name:** AecLayoutGrid3DZNodes

**Create using:** N/A

**Access via:** AecLayoutGrid3D.ZNodes

You can access the members of this collection, but they contain no useful information.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Methods</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Count</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Item</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LayoutGrid3D</td>
<td></td>
</tr>
</tbody>
</table>
Layout Node Object

Provides common properties shared by all types of layout nodes.

**VBA object name:** AecLayoutNode

**Create using:** N/A

**Access via:** AecLayoutNodes.Item

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Spacing</td>
<td>None</td>
</tr>
</tbody>
</table>
Collection of layout nodes.

**VBA object name:** AecLayoutNodes

**Create using:** N/A

**Access via:** AecLayoutCurve.Nodes

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>Count</td>
</tr>
</tbody>
</table>
Provides common functionality for layout curve and layout grid objects.

VBA object name: AecLayoutTool

Create using: N/A

Access via:
- ModelSpace.Item
- PaperSpace.Item
- Block.Item
- SelectionSet.Item
- Group.Item

AecLayoutTool can be used as a generic reference to layout objects.

Methods
- ArrayPolar
- ArrayRectangular
- AttachAnchor

Properties
- Application
ClosestNode
Copy
Delete
GetAnchor
GetBoundingBox
GetExtensionDictionary
GetXData
Highlight
IntersectWith
Mirror
Mirror3D
Move
NodeLocation
ReleaseAnchor
Rotate
Rotate3D
ScaleEntity
SetXData
TransformBy
Update

Color
Description
Document
Handle
HasExtensionDictionary
Hyperlinks
Layer

Linetype
LinetypeScale
Lineweight
Location
Normal
ObjectID
ObjectName
OwnerID
PlotStyleName
Rotation
TrueColor
Visible

Events
Modified
An instance of a mask block, a two-dimensional block that covers part of an AEC object in a plan (2D) view.

**VBA object name:** AecMaskBlockRef

**Create using:** ModelSpace.AddCustomObject("AecMaskBlockRef")

**Access via:**
- ModelSpace.Item
- PaperSpace.Item
- Block.Item
- SelectionSet.Item
- Group.Item

**Properties**
- Application
- Color
- CutProfile

**Methods**
- ArrayPolar
<table>
<thead>
<tr>
<th>Description</th>
<th>Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrayRectangular</td>
<td>AttachAnchor</td>
</tr>
<tr>
<td>Copy</td>
<td>Delete</td>
</tr>
<tr>
<td>GetAnchor</td>
<td>GetBoundingBox</td>
</tr>
<tr>
<td>GetExtensionDictionary</td>
<td>Hyperlinks</td>
</tr>
<tr>
<td>GetXData</td>
<td>Linetype</td>
</tr>
<tr>
<td>Highlight</td>
<td>LinetypeScale</td>
</tr>
<tr>
<td>IntersectWith</td>
<td>Lineweight</td>
</tr>
<tr>
<td>Mirror</td>
<td>Location</td>
</tr>
<tr>
<td>Mirror3D</td>
<td>Normal</td>
</tr>
<tr>
<td>Move</td>
<td>ObjectID</td>
</tr>
<tr>
<td>ReleaseAnchor</td>
<td>ObjectName</td>
</tr>
<tr>
<td>Rotate</td>
<td>OwnerID</td>
</tr>
<tr>
<td>Rotate3D</td>
<td>PlotStyleName</td>
</tr>
<tr>
<td>ScaleEntity</td>
<td>Rotation</td>
</tr>
<tr>
<td>SetXData</td>
<td>ScaleX</td>
</tr>
<tr>
<td>TransformBy</td>
<td>ScaleY</td>
</tr>
<tr>
<td>Update</td>
<td>ScaleZ</td>
</tr>
<tr>
<td></td>
<td>Style</td>
</tr>
</tbody>
</table>
StyleName

TrueColor

Visible
Mask Block Style

Defines the properties of a mask block object.

**VBA object name:** AecMaskBlockStyle

**Create using:** AecMaskBlockStyles.Add("Name")

**Access via:** AecMaskBlockStyles.Item

Most mask block style properties are not yet exposed through ActiveX.

### Properties

- **AlternateName**
- **Application**

### Methods

- **Delete**
- **Document**

### Events
<table>
<thead>
<tr>
<th>Method</th>
<th>Data Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetExtensionDictionary</td>
<td>Handle</td>
<td></td>
</tr>
<tr>
<td>GetXData</td>
<td>HasExtensionDictionary</td>
<td></td>
</tr>
<tr>
<td>SetXData</td>
<td>Name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ObjectID</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ObjectName</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OwnerID</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modified</td>
<td></td>
</tr>
</tbody>
</table>
The collection of mask block styles in the specified drawing.

**VBA object name:** AecMaskBlockStyles

**Create using:** N/A

**Access via:** AecBaseDatabase.MaskBlocks

Use of this collection is mostly limited to listing the name and description of the styles.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Application</td>
</tr>
<tr>
<td>Delete</td>
<td>Count</td>
</tr>
<tr>
<td>GetExtensionDictionary</td>
<td>Document</td>
</tr>
<tr>
<td>GetXData</td>
<td>Handle</td>
</tr>
</tbody>
</table>

**Events**
<table>
<thead>
<tr>
<th></th>
<th>Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Has</td>
<td>HasExtensionDictionary</td>
</tr>
<tr>
<td>Item</td>
<td>ObjectID</td>
</tr>
<tr>
<td>Remove</td>
<td>ObjectName</td>
</tr>
<tr>
<td>SetXData</td>
<td>OwnerID</td>
</tr>
</tbody>
</table>
An object that has behaviors based on its shape.

**VBA object name:** AecMassElement

**Create using:** AddCustomObject("AecMassElement")

**Access via:**
- ModelSpace.Item
- PaperSpace.Item
- Block.Item
- SelectionSet.Item
- Group.Item

**Properties**

- Application
- Color
- Depth
Methods

- ArrayPolar
- ArrayRectangular
- AttachAnchor
- Copy
- Delete
- ExportFreeForm
- GetAnchor
- GetBoundingBox
- GetExtensionDictionary
- GetXData
- Highlight
- ImportFreeForm
- IntersectWith
- Mirror
- Mirror3D
- Move
- ReleaseAnchor

Description

Deviation

Document

Handle

HasExtensionDictionary

Height

Hyperlinks

Layer

Linetype

LinetypeScale

Lineweight

Location

MassGroup

MassGroupName

Normal

Modified

ObjectID

ObjectName

Operation

OwnerID

PlotStyleName
<table>
<thead>
<tr>
<th>Function</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rotate</td>
<td>Profile</td>
</tr>
<tr>
<td>Rotate3D</td>
<td>ProfileStyle</td>
</tr>
<tr>
<td>ScaleEntity</td>
<td>ProfileStyleName</td>
</tr>
<tr>
<td>SetXData</td>
<td>Radius</td>
</tr>
<tr>
<td>TransformBy</td>
<td>Rise</td>
</tr>
<tr>
<td>Update</td>
<td>Rotation</td>
</tr>
<tr>
<td></td>
<td>Style</td>
</tr>
<tr>
<td></td>
<td>StyleName</td>
</tr>
<tr>
<td></td>
<td>Subtype</td>
</tr>
<tr>
<td></td>
<td>TrueColor</td>
</tr>
<tr>
<td></td>
<td>Type</td>
</tr>
<tr>
<td></td>
<td>Visible</td>
</tr>
<tr>
<td></td>
<td>Volume</td>
</tr>
<tr>
<td></td>
<td>Width</td>
</tr>
</tbody>
</table>
Object

Defines the properties of a mass element object.

**VBA object name:** AecMassElementStyle

**Create using:** AecMassElementStyles.Add("Name")

**Access via:** AecMassElementStyles.Item

<table>
<thead>
<tr>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlternateName</td>
</tr>
<tr>
<td>Application</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete</td>
</tr>
<tr>
<td>Description</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GetExtensionDictionary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handle</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modified</td>
</tr>
<tr>
<td>GetXData</td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td>SetXData</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Collection

The collection of mass element styles in the specified drawing.

VBA object name: AecMassElementStyles

Create using: N/A

Access via: AecBaseDatabase.MassElementStyles

Methods

Add
Delete
GetExtensionDictionary
GetXData
Has

Properties

Application
Count
Document
Handle

Events

Modified
HasExtensionDictionary
<table>
<thead>
<tr>
<th>Item</th>
<th>HasExtensionDictionary</th>
<th>Item</th>
<th>HasExtensionDictionary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td></td>
<td>Item</td>
<td></td>
</tr>
<tr>
<td>Remove</td>
<td>ObjectId</td>
<td>Rename</td>
<td>ObjectName</td>
</tr>
<tr>
<td>Rename</td>
<td></td>
<td>SetXData</td>
<td>OwnerID</td>
</tr>
</tbody>
</table>
Mass Group Object

A group of mass elements combined to form a complex shape.

**VBA object name:** AecMassGroup

**Create using:** AecMassGroups.Add

**Access via:** AecMassGroups.Item

### Properties
- Application
- Color
- Description
- Document
- Handle
- HasExtensionDictionary
<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete</td>
<td>Hyperlinks</td>
</tr>
<tr>
<td>ExportFreeForm</td>
<td>Layer</td>
</tr>
<tr>
<td>GetAnchor</td>
<td>Linetype</td>
</tr>
<tr>
<td>GetBoundingBox</td>
<td>LinetypeScale</td>
</tr>
<tr>
<td>GetExtensionDictionary</td>
<td>Lineweight</td>
</tr>
<tr>
<td>GetXData</td>
<td>Location</td>
</tr>
<tr>
<td>Highlight</td>
<td>MassGroup</td>
</tr>
<tr>
<td>IntersectWith</td>
<td>MassGroupName</td>
</tr>
<tr>
<td>Mirror</td>
<td>Name</td>
</tr>
<tr>
<td>Mirror3D</td>
<td>Normal</td>
</tr>
<tr>
<td>Move</td>
<td>ObjectID</td>
</tr>
<tr>
<td>ReleaseAnchor</td>
<td>ObjectName</td>
</tr>
<tr>
<td>Rotate</td>
<td>Operation</td>
</tr>
<tr>
<td>Rotate3D</td>
<td>OwnerID</td>
</tr>
<tr>
<td>ScaleEntity</td>
<td>PlotStyleName</td>
</tr>
<tr>
<td>SetXData</td>
<td>Rotation</td>
</tr>
<tr>
<td>TransformBy</td>
<td>SupportsAnchoring</td>
</tr>
<tr>
<td>Update</td>
<td>TrueColor</td>
</tr>
<tr>
<td></td>
<td>Visible</td>
</tr>
<tr>
<td></td>
<td>Volume</td>
</tr>
</tbody>
</table>
The collection of mass groups in the specified drawing.

**VBA object name:** AecMassGroups

**Create using:** N/A

**Access via:** AecBaseDatabase.MassGroups

**Properties**
- Count
- Database
AecMVBlockDisplayRepDef Object

A display representation definition for a multi-view block style.

**VBA object name:** AecMVBlockDisplayRepDef

**Create using:** N/A

**Access via:** AecMVBlockDisplayRepDefs.Item

**Properties**

- **DisplayName**

- **Index**

- **Owner**

- **ViewBlockDefs**
**AecMVBlockDisplayRepDefs Collection**

The collection of display representation definitions for a multi-view block style.

VBA object name: AecMVBlockDisplayRepDefs

Create using: N/A

Access via: AecMVBlockStyle.DisplayRepDefs

- **Properties**: Count, Owner
- **Methods**: Item, Count, Events
Multi-View Block

An instance of a block that can have different representations in different view directions.

**VBA object name:** AecMVBlockRef

**Create using:**
```
AddCustomObject("AecMVBlockRef")
```

**Access via:**
- ModelSpace.Item
- PaperSpace.Item
- Block.Item
- SelectionSet.Item
- Group.Item

**Properties**

**Application**

**Color**

**Methods**

**Description**
Update

TrueColor

ViewBlocks

Visible
Multi-View Block

**Style Object**

Defines the properties of a multi-view block object.

**VBA object name:** AecMVBlockStyle

**Create using:**

```
AecMVBlockStyles.Add("Name")
```

**Access via:**

```
AecMVBlockStyles.Item
```

Most multi-view block style properties are not yet exposed through ActiveX.

### Properties

- **AlternateName**
- **Application**
- **Description**

### Methods

- **DisplayRepDefs**
<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete</td>
<td>Document</td>
<td></td>
</tr>
<tr>
<td>GetExtensionDictionary</td>
<td>Handle</td>
<td>Modified</td>
</tr>
<tr>
<td>GetXData</td>
<td>HasExtensionDictionary</td>
<td></td>
</tr>
<tr>
<td>SetXData</td>
<td>InterferenceBlockName</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ObjectId</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ObjectName</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OwnerID</td>
<td></td>
</tr>
</tbody>
</table>
The collection of multi-view block styles in the specified drawing.

**VBA object name:** AecMVBlockStyles

**Create using:** N/A

**Access via:** AecBaseDatabase.MVBlockStyles

Use of this collection is mostly limited to listing the name and description of the styles.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Application</td>
</tr>
<tr>
<td>Delete</td>
<td>Count</td>
</tr>
<tr>
<td>GetExtensionDictionary</td>
<td>Document</td>
</tr>
<tr>
<td>GetXData</td>
<td>Handle</td>
</tr>
</tbody>
</table>

**Events**
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Has</td>
<td>HasExtensionDictionary</td>
</tr>
<tr>
<td>Item</td>
<td>ObjectId</td>
</tr>
<tr>
<td>Remove</td>
<td>ObjectName</td>
</tr>
<tr>
<td>SetXData</td>
<td>OwnerID</td>
</tr>
<tr>
<td>Modified</td>
<td></td>
</tr>
</tbody>
</table>
**AecObject Object**

Provides common functionality for AEC database objects.

**VBA object name:** AecObject

**Create using:** N/A

**Access via:**
- ModelSpace.Item
- PaperSpace.Item
- Block.Item
- SelectionSet.Item
- Group.Item

Although AecObject can be used as a generic reference to AEC objects, it is better to use AecGeo as a generic reference. AecGeo is the greatest common dominator for all AEC objects.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>Delete</td>
<td>Document</td>
</tr>
<tr>
<td>GetExtensionDictionary</td>
<td>Handle</td>
</tr>
<tr>
<td>GetXData</td>
<td>HasExtensionDictionary</td>
</tr>
<tr>
<td>SetXData</td>
<td>ObjectId</td>
</tr>
<tr>
<td></td>
<td>ObjectName</td>
</tr>
<tr>
<td></td>
<td>OwnerID</td>
</tr>
</tbody>
</table>
An Aec Polygon object.

VBA object name: AecPolygon

Create using: ModelSpace.AddCustomObject("AecPolygon")

Access via:
- ModelSpace.Item
- PaperSpace.Item
- Block.Item
- SelectionSet.Item
- Group.Item

You can modify existing AECPolygon objects. Although you can create AECPolygons, you cannot control the number or length of its sides, so it's not very useful.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrayRectangular</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>AttachAnchor</td>
<td>Document</td>
<td></td>
</tr>
<tr>
<td>Copy</td>
<td>Handle</td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td>HasExtensionDictionary</td>
<td></td>
</tr>
<tr>
<td>GetAnchor</td>
<td>Hyperlinks</td>
<td></td>
</tr>
<tr>
<td>GetBoundingBox</td>
<td>Layer</td>
<td></td>
</tr>
<tr>
<td>GetExtensionDictionary</td>
<td>Linetype</td>
<td></td>
</tr>
<tr>
<td>GetXData</td>
<td>LinetypeScale</td>
<td>Events</td>
</tr>
<tr>
<td>Highlight</td>
<td>Lineweight</td>
<td>Modified</td>
</tr>
<tr>
<td>IntersectWith</td>
<td>Location</td>
<td></td>
</tr>
<tr>
<td>Mirror</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>Mirror3D</td>
<td>ObjectID</td>
<td></td>
</tr>
<tr>
<td>Move</td>
<td>ObjectName</td>
<td></td>
</tr>
<tr>
<td>ReleaseAnchor</td>
<td>OwnerID</td>
<td></td>
</tr>
<tr>
<td>Rotate</td>
<td>PlotStyleName</td>
<td></td>
</tr>
<tr>
<td>Rotate3D</td>
<td>Profile</td>
<td></td>
</tr>
<tr>
<td>ScaleEntity</td>
<td>Rotation</td>
<td></td>
</tr>
<tr>
<td>SetXData</td>
<td>Style</td>
<td></td>
</tr>
<tr>
<td>TransformBy</td>
<td>StyleName</td>
<td></td>
</tr>
<tr>
<td>Update</td>
<td>TrueColor</td>
<td></td>
</tr>
</tbody>
</table>
Visible
**AEC Polygon Style**

Defines the properties of an Aec Polygon object.

**VBA object name:** AecPolygonStyle

**Create using:**
AecPolygonStyles.Add("Name")

**Access via:**
AecPolygonStyles.Item

<table>
<thead>
<tr>
<th>Properties</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AlternateName</td>
<td></td>
</tr>
<tr>
<td>Application</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Document</td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td></td>
</tr>
<tr>
<td>EdgeWidth</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Parameters</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>GetExtensionDictionary</td>
<td></td>
</tr>
<tr>
<td>EdgeWidthJustify</td>
<td></td>
</tr>
<tr>
<td>GetXData</td>
<td>Handle</td>
</tr>
<tr>
<td>SetXData</td>
<td>HasExtensionDictionary</td>
</tr>
<tr>
<td></td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td>ObjectId</td>
</tr>
<tr>
<td></td>
<td>ObjectName</td>
</tr>
<tr>
<td></td>
<td>OwnerID</td>
</tr>
<tr>
<td>Events</td>
<td>Modified</td>
</tr>
</tbody>
</table>
AEC Polygon Styles Collection

The collection of Aec Polygon styles in the specified drawing.

**VBA object name:** AecPolygonStyles

**Create using:** N/A

**Access via:** AecBaseDatabase.PolygonStyles

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Application</td>
</tr>
<tr>
<td>Delete</td>
<td>Count</td>
</tr>
<tr>
<td>GetExtensionDictionary</td>
<td>Document</td>
</tr>
<tr>
<td>GetXData</td>
<td>Handle</td>
</tr>
<tr>
<td>Has</td>
<td>HasExtensionDictionary</td>
</tr>
</tbody>
</table>

**Events**

Modified
<table>
<thead>
<tr>
<th>Item</th>
<th>ObjectID</th>
<th>ObjectName</th>
<th>OwnerID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetXData</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Profile Object

Defines the properties of a profile style.

**VBA object name:** AecProfile

**Create using:** New AecProfile

**Access via:** AecProfileStyle.Profile

<table>
<thead>
<tr>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
</tr>
<tr>
<td>Clear</td>
</tr>
<tr>
<td>CopyFrom</td>
</tr>
<tr>
<td>Intersect</td>
</tr>
<tr>
<td>ScaleXY</td>
</tr>
</tbody>
</table>

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bounds</td>
</tr>
<tr>
<td>Centroid</td>
</tr>
<tr>
<td>Coordinates</td>
</tr>
<tr>
<td>Rings</td>
</tr>
<tr>
<td>Perimeter</td>
</tr>
<tr>
<td>Subtract</td>
</tr>
</tbody>
</table>
Profile Style Object

A custom shape defined by two-dimensional closed polylines.

**VBA object name**: AecProfileStyle

**Create using**: AecProfileStyles.Add("Name")

**Access via**: AecProfileStyles.Item

**Note**: Profiles are inserted into drawings as polylines, not as AEC objects.

<table>
<thead>
<tr>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlternateName</td>
</tr>
<tr>
<td>Application</td>
</tr>
<tr>
<td>Description</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete</td>
</tr>
<tr>
<td>Document</td>
</tr>
<tr>
<td>Handle</td>
</tr>
</tbody>
</table>
GetExtensionDictionary
GetXData
SetXData

Handle
HasExtensionDictionary
Modified

Name

ObjectID

ObjectName

OwnerID

Profile
Profile Styles

Collection

The collection of profile styles in the specified drawing.

VBA object name: AecProfileStyles

Create using: N/A

Access via: AecBaseDatabase.ProfileStyles

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Application</td>
</tr>
<tr>
<td>Delete</td>
<td>Count</td>
</tr>
<tr>
<td>GetExtensionDictionary</td>
<td>Document</td>
</tr>
<tr>
<td>GetXData</td>
<td>Handle</td>
</tr>
<tr>
<td>Has</td>
<td>HasExtensionDictionary</td>
</tr>
</tbody>
</table>

Events

Modified
<table>
<thead>
<tr>
<th>Item</th>
<th>ObjectID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove</td>
<td>ObjectName</td>
</tr>
<tr>
<td>SetXData</td>
<td>OwnerID</td>
</tr>
</tbody>
</table>
A closed loop of polyline segments, often used as a group inside an AecProfile object.

**VBA object name:** AecRing

**Create using:** AecRings.Add

**Access via:** AecRings.Item

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear</td>
<td>Coordinates</td>
</tr>
<tr>
<td>Contains</td>
<td>Index</td>
</tr>
<tr>
<td>CopyFrom</td>
<td>Profile</td>
</tr>
<tr>
<td>FromPoints</td>
<td>Valid</td>
</tr>
<tr>
<td>FromPolyline</td>
<td>Void</td>
</tr>
</tbody>
</table>
The collection of rings in an AecProfile object.

**VBA object name:** AecRings

**Create using:** N/A

**Access via:** AecProfile.Rings

**Methods**
- Add
- Item
- Remove

**Properties**
- Count
- Profile
Slice of a massing group or any AEC object, typically used to generate floor plates and space boundaries.

**VBA object name:** AecSlice

**Create using:** AddCustomObject("AecSlice")

**Access via:**
- ModelSpace.Item
- PaperSpace.Item
- Block.Item
- SelectionSet.Item
- Group.Item

**Methods**
- ArrayPolar
- ArrayRectangular
- AttachAnchor

**Properties**
- Application
- Area
- Color
<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
<th>Event</th>
<th>Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClosestNode</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Copy</td>
<td>Document</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td>Handle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetAnchor</td>
<td>HasExtensionDictionary</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetBoundingBox</td>
<td>Hyperlinks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetExtensionDictionary</td>
<td>Layer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetXData</td>
<td>Linetype</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Highlight</td>
<td>LinetypeScale</td>
<td>Events</td>
<td>Modified</td>
</tr>
<tr>
<td>IntersectWith</td>
<td>Lineweight</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mirror</td>
<td>Location</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mirror3D</td>
<td>Normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Move</td>
<td>ObjectID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NodeLocation</td>
<td>ObjectName</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReleaseAnchor</td>
<td>OwnerID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rotate</td>
<td>PlaneDepth</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rotate3D</td>
<td>PlaneWidth</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ScaleEntity</td>
<td>PlotStyleName</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetXData</td>
<td>Profile</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TransformBy</td>
<td>Rotation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Update</td>
<td>TrueColor</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Visible
Aec View Block Object

View block used for display representations by a multi-view block.

**VBA object name:** AecViewBlock

**Create using:** N/A

**Access via:** AecViewBlocks.Item

<table>
<thead>
<tr>
<th>Properties</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Index</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>MVBlockRef</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

**Name**

**XOffset**

**YOffset**
ZOffset
Aec View Blocks

Collection

A collection of view blocks used for display representations by a multi-view block.

VBA object name: AecViewBlocks

Create using: N/A

Access via: AecMVBlockRef.ViewBlocks

You cannot add new view blocks.

Properties

Methods

Item

Count

MVBlockRef
**Object**

A view block definition for a multi-view block display representation definition.

**VBA object name:** AecViewBlockDef

**Create using:** N/A

**Access via:** AecViewBlockDefs.Item

**Methods**

- **Owner**

**Events**

- **ViewDirection**
**AecViewBlockDefs**

**Collection**

The collection of view block definitions for a multi-view block display representation definition.

**VBA object name:** AecViewBlockDefs

**Create using:** N/A

**Access via:** AecMVBlockDisplayRepDef.ViewBlockDefs

**Methods**

- Add
- Item
- Remove

**Properties**

- Count
- Owner

**Events**

- Events
Volume Layout Tool

Object

Provides standard functionality for all 3D layout tools.

**VBA object name:** AecVolumeLayoutTool

**Create using:** N/A

**Access via:**
- ModelSpace.Item
- PaperSpace.Item
- Block.Item
- SelectionSet.Item
- Group.Item

AecVolumeLayoutTool can be used as a generic reference to 3D layout (volume) grid objects, but you would normally use [AecLayoutGrid3D](#).

**Methods**

- ArrayPolar
- ArrayRectangular

**Properties**
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttachAnchor</td>
<td>Application</td>
</tr>
<tr>
<td>ClosestNode</td>
<td>Color</td>
</tr>
<tr>
<td>Copy</td>
<td>Description</td>
</tr>
<tr>
<td>Delete</td>
<td>Document</td>
</tr>
<tr>
<td>GetAnchor</td>
<td>Handle</td>
</tr>
<tr>
<td>GetBoundingBox</td>
<td>HasExtensionDictionary</td>
</tr>
<tr>
<td>GetExtensionDictionary</td>
<td>Hyperlinks</td>
</tr>
<tr>
<td>GetXData</td>
<td>Layer</td>
</tr>
<tr>
<td>Highlight</td>
<td>Linetype</td>
</tr>
<tr>
<td>IntersectWith</td>
<td>LinetypeScale</td>
</tr>
<tr>
<td>Mirror</td>
<td>Lineweight</td>
</tr>
<tr>
<td>Mirror3D</td>
<td>Location</td>
</tr>
<tr>
<td>Move</td>
<td>Normal</td>
</tr>
<tr>
<td>NodeLocation</td>
<td>ObjectId</td>
</tr>
<tr>
<td>ReleaseAnchor</td>
<td>ObjectName</td>
</tr>
<tr>
<td>Rotate</td>
<td>OwnerID</td>
</tr>
<tr>
<td>Rotate3D</td>
<td>PlotStyleName</td>
</tr>
<tr>
<td>ScaleEntity</td>
<td>Rotation</td>
</tr>
<tr>
<td>SetXData</td>
<td>TrueColor</td>
</tr>
<tr>
<td>TransformBy</td>
<td>Visible</td>
</tr>
</tbody>
</table>
Update
All Drawing Objects

All of these objects can be defined as an AcadEntity object.

<table>
<thead>
<tr>
<th>AecAnchorEntToGridAssembly</th>
<th>AecAnchor</th>
<th>AecAnchorEntToLayoutCell</th>
<th>AecAnchorEntToLayoutNode</th>
<th>AecAnchorEntToLayoutVolume</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AecAnchor</strong></td>
<td></td>
<td><strong>AecAnchorEntTo</strong></td>
<td><strong>AecAnchorEntTo</strong></td>
<td><strong>AecAnchorEntTo</strong></td>
</tr>
<tr>
<td>AecAnchorLeadEntToNode</td>
<td>AecAnchorToRef</td>
<td>AecAnchorEntToLayoutNode</td>
<td>AecAnchorEntToLayoutNode</td>
<td>AecAnchorEntToLayoutNode</td>
</tr>
<tr>
<td>AecCamera</td>
<td>AecAnchorToRef</td>
<td>AecAnchorEntToLayoutCell</td>
<td>AecAnchorEntToLayoutCell</td>
<td>AecAnchorEntToLayoutCell</td>
</tr>
<tr>
<td>AecClipVolRes</td>
<td>AecAnchorToRef</td>
<td>AecAnchorEntToLayoutNode</td>
<td>AecAnchorEntToLayoutNode</td>
<td>AecAnchorEntToLayoutNode</td>
</tr>
<tr>
<td><strong>AecEntity</strong></td>
<td>AecAnchorToRef</td>
<td>AecAnchorEntToLayoutCell</td>
<td>AecAnchorEntToLayoutCell</td>
<td>AecAnchorEntToLayoutCell</td>
</tr>
<tr>
<td>AecGridAssembly</td>
<td>AecAnchorToRef</td>
<td>AecAnchorEntToLayoutCell</td>
<td>AecAnchorEntToLayoutCell</td>
<td>AecAnchorEntToLayoutCell</td>
</tr>
<tr>
<td>AecLayerKeyStyle</td>
<td>AecAnchorToRef</td>
<td>AecAnchorEntToLayoutCell</td>
<td>AecAnchorEntToLayoutCell</td>
<td>AecAnchorEntToLayoutCell</td>
</tr>
<tr>
<td>AecMaskBlockRef</td>
<td>AecAnchorToRef</td>
<td>AecAnchorEntToLayoutCell</td>
<td>AecAnchorEntToLayoutCell</td>
<td>AecAnchorEntToLayoutCell</td>
</tr>
<tr>
<td>AecMassElement</td>
<td>AecAnchorToRef</td>
<td>AecAnchorEntToLayoutCell</td>
<td>AecAnchorEntToLayoutCell</td>
<td>AecAnchorEntToLayoutCell</td>
</tr>
<tr>
<td>AecMVBlockStyles</td>
<td>AecAnchorToRef</td>
<td>AecAnchorEntToLayoutCell</td>
<td>AecAnchorEntToLayoutCell</td>
<td>AecAnchorEntToLayoutCell</td>
</tr>
<tr>
<td>AecPolygonStyle</td>
<td>AecAnchorToRef</td>
<td>AecAnchorEntToLayoutCell</td>
<td>AecAnchorEntToLayoutCell</td>
<td>AecAnchorEntToLayoutCell</td>
</tr>
<tr>
<td>AecProfileStyles</td>
<td>AecAnchorToRef</td>
<td>AecAnchorEntToLayoutCell</td>
<td>AecAnchorEntToLayoutCell</td>
<td>AecAnchorEntToLayoutCell</td>
</tr>
</tbody>
</table>
Methods

Add Adds an object to a collection, or adds two AecProfile objects.

ArrayPolar Creates a polar array of objects given a NumberOfObjects, AngleToFill, and CenterPoint.

ArrayRectangular Creates a 2D or 3D rectangular array of objects.

AttachAnchor Attaches an anchor to an AEC Entity

AttachEntity Anchors one AecGeo object to another.

Clear Clears AecRings and AecProfiles of segments or rings.

ClosestNode Returns the index of the node closest to a given point.

Contains Tests to see if a ring is inside another ring.

ConvertToCurrentAreaDisplay Converts an area value using the LinearUnit and AreaDisplayUnit settings.

ConvertToCurrentVolumeDisplay Converts a volume value using the LinearUnit and VolumeDisplayUnit settings.

Copy Duplicates the given object to the same location.

CopyFrom Copies AecRings and AecProfiles.
CopyObjects Duplicates multiple objects (deep cloning).

Delete Deletes the specified object.

ExpandKey Calculates and returns the Layer Name that would be generated given the supplied 'Key'.

ExportFreeForm Exports a Free Form representation to a file.

FromPoints Makes a ring from 2D Points.

FromPolyline Makes a ring from a 2D Polyline.

GenerateLayer Generate the layer record, given a Layer Key.

GetAnchor Returns the anchor attached to an object.

GetBoundingBox Gets two points of a box enclosing the specified object.

GetExtensionDictionary Gets the extension dictionary associated with an object.

GetXdata Gets the extended data (XData) associated with an object.

HandleToObject Gets the object that corresponds to the given handle.

Has Determines if an object is a member of the specified collection.

Highlight Sets the highlight status for the given object, or for all objects in a given selection set.
**ImportFreeForm** Imports a Free Form representation from a file.

**Init** Initialization function which must be called before the AEC database object can be accessed

**Intersect** Boolean Intersects an AecProfile with the current AecProfile.

**IntersectWith** Gets the points where one object intersects another object in the drawing.

**Item** Returns a member object of a collection.

**Mirror** Creates a mirror image copy of a planar object around an axis.

**Mirror3d** Creates a mirror image of the given object about a plane.

**Move** Moves an object along a vector.

**NodeLocation** Returns the location of the node at a given index.

**ObjectIdToObject** Gets the object that corresponds to the given object ID.
ReleaseAnchor Removes an anchor from an object.
Remove Removes an item from a collection.
Rename Renames an item in an AecDictionary.
Rotate Rotates an object around a base point.
Rotate3d Rotates an object around a 3D axis. Point1 and Point2 define the line that becomes the axis of rotation.

S

SaveAsDefault Saves the DatabasePreferences to the registry.
ScaleEntity Scales an object equally in the
ScaleXY Scales a profile in the X and Y directions.
SetXdata Sets the extended data (XData) associated with an object.
Subtract Subtracts an AecProfile from the current AecProfile.

T

TransformBy Moves, scales, or rotates an object given a 4x4 transformation matrix.

U

Update Updates the object in the drawing window.

V

W

X

Y

Z
Add Method

Adds an object to a collection, or adds two AecProfile objects.

See Also | Example

Signatures: Overview

Layer Keys,
Layer Key Styles,
Mask Block Styles,
Mass Element Styles,
Multi-view Block Styles,
Polygon Styles,
Profile Styles,
Display Configurations,
Display Sets

RetVal = collection.Add(Name)

Profiles

object.Add(profile)

Rings

RetVal = collection.Add()

ViewBlockDefs

RetVal = collection.Add(AcadBlock)

Signatures: Detail

RetVal = collection.Add(Name)
collection

AecLayerKeys, AecLayerKeyStyles, AecMassElementStyles, AecMaskBlockStyles, AecMVBlockStyles, AecPolygonStyles, AecProfileStyles, AecDisplayConfigurations, AecDisplaySets, AecViewBlockDefs
The collection the object is being added to.

Name

String; input-only
Name of the object to be added to the collection.

RetVal

AecLayerKey, AecLayerKeyStyle, AecMaskBlockStyle, AecMassElementStyle, AecMVBlockStyle, AecPolygonStyle, AecProfileStyle, AecDisplayConfigurations, AecDisplaySets, AecViewBlockDefs
The newly added object.

object.Add(profile)

object

AecProfile
The profile you are adding a new profile object to.

profile

AecProfile; input-only
AecProfile to add to the first AecProfile

RetVal = collection.Add()

collection

AecRings
The collection you are adding a new ring definition to.
RetVal

AecRing
The newly added object.

RetVal = collection.Add(AcadBlock)

collection

AecViewBlockDefs
The collection you are adding a new block to.

AcadBlock

AcadBlock object; input-only
The block to add to the collection.

RetVal

AecViewBlockDef
The newly created view block definition object.
Append Method

Appends an existing display representation to the collection.

See Also | Example

Signature

object.Append DisplayRepresentation

Object

AecDisplayRepresentations The object or objects this method applies to.

DisplayRepresentation

AecDisplayRepresentation; input-only
The display representation to add to the collection.

Remarks

This method applies to a display representations collection in a display set.
ArrayPolar Method

Creates a polar array of objects given a NumberOfObjects, AngleToFill, and CenterPoint.

See Also | Example

Signature

RetVal = object.ArrayPolar (NumberOfObjects, AngleToFill, CenterPoint)

Object


The object or objects this method applies to.

NumberOfObjects

Integer; input-only

The number of objects to be created in the polar array. This must be a positive integer greater than 1.

AngleToFill

Double; input-only

The angle to fill in radians. A positive value specifies counterclockwise rotation. A negative value specifies clockwise rotation. An error is returned for an angle that equals 0.

CenterPoint

Variant (three-element array of doubles); input-only

The 3D WCS coordinates specifying the center point for the polar array.
RetVal

Variant Array (array of objects)
The array of new objects.

Remarks

AutoCAD determines the distance from the array's center point to a reference point on the last object selected. The reference point used depends on the type of object previously selected. AutoCAD uses the center point of a circle or arc, the insertion point of a block or shape, the start point of text, and one endpoint of a line or trace.

\[
\begin{align*}
(0,0,0) & \quad \Rightarrow \\
(0,0,0) & \quad \Rightarrow
\end{align*}
\]

*Polar array with NumberOfObjects = 5, AngleToFill = 180, CenterPoint = 0,0,0.*

NOTE You cannot execute this method while simultaneously iterating through a collection. An iteration will open the work space for a read-only operation, while this method attempts to perform a read-write operation. Complete any iteration before you call this method.
ArrayRectangular Method

Creates a 2D or 3D rectangular array of objects.

See Also | Example

Signature

RetVal = object.ArrayRectangular (NumberOfRows, NumberOfColumns, NumberOfLevels, DistBetweenRows, DistBetweenColumns, DistBetweenLevels)

Object


The object or objects this method applies to.

NumberOfRows

Integer; input-only
The number of rows in the rectangular array. This must be a positive number. If this number is 1, then NumberOfColumns must be greater than 1.

NumberOfColumns

Integer; input-only
The number of columns in the rectangular array. This must be a positive number. If this number is 1, then NumberOfRows must be greater than 1.

NumberOfLevels

Integer; input-only
The number of levels in a 3D array.
DistBetweenRows
Double; input-only
The distance between the rows. If the distance between rows is a positive number, rows are added upward from the base entity. If the distance is a negative number, rows are added downward.

DistBetweenColumns
Double; input-only
The distance between the columns. If the distance between columns is a positive number, columns are added to the right of the base entity. If the distance is a negative number, columns are added to the left.

DistBetweenLevels
Double; input-only
The distance between the array levels. If the distance between levels is a positive number, levels are added in the positive direction from the base entity. If the distance is a negative number, levels are added in the negative direction.

RetVal
Variant Array (array of objects)
The array of newly created objects.

Remarks
For a 2D array, specify the NumberOfRows, NumberOfColumns, DistBetweenRow, and DistBetweenColumns. For creating a 3D array, specify the NumberOfLevels and DistBetweenLevels as well.

A rectangular array is constructed by replicating the object in the selection set the appropriate number of times. If you define one row, you must specify more than one column and vice versa.

The object in the selection set is assumed to be in the lower left-hand corner, and the array is generated up and to the right. If the distance between rows is a negative number, rows are added downward. If the
distance between columns is a negative number, the columns are added to the left.

AutoCAD builds the rectangular array along a baseline defined by the current snap rotation angle.

\[
\begin{array}{ccc}
\text{a} & \text{a} & \text{a} \\
\text{a} & \text{a} & \text{a} \\
\text{a} & \text{a} & \text{a} \\
\text{b} & \text{a} & \text{b} \\
\end{array}
\]

Rectangular array with NumberOfRows = 4, NumberOfColumns = 3, DistBetweenRows = a, DistBetweenColumns = b. The base entity is represented in blue.

NOTE You cannot execute this method while simultaneously iterating through a collection. An iteration will open the work space for a read-only operation, while this method attempts to perform a read-write operation. Complete any iteration before you call this method.
AttachAnchor Method

Attaches an anchor to an AEC Entity.

See Also | Example

Signature

object.AttachAnchor anchor

object


The object this method applies to.

anchor

AecAnchor; The anchor that will be attached
This can be any of the AEC Anchor objects.
**AttachEntity Method**

Anchors an AEC entity to an AutoCAD or AEC entity.

See Also | Example

**Signature**

object.AttachEntity entity

object

AecAnchor, AecAnchorEntToCurve, AecAnchorEntToLayoutCell, AecAnchorEntToLayoutNode, AecAnchorEntToLayoutVolume, AecAnchorExtendedTagToEnt AecAnchorLeadEntToNode, AecAnchorToRef

The object this method applies to.

entity

AecEntity; input-only

The AEC entity to be attached to the anchored object.

**Remarks**

Although AecAnchorEntToGridAssembly inherits the AttachEntity method from AecAnchor, it is disabled.
Clear Method

Clears AecRings and AecProfiles of segments or rings.

See Also | Example

Signature

object.Clear()

object

AecRing, AecProfile

The object to be cleared.
ClosestNode Method

Returns the index of the node closest to a given point.

See Also | Example

Signature

RetVal = object.ClosestNode(Point)

Object

AecCellLayoutTool, AecClipVol, AecGridAssembly,
AecLayoutCurve, AecLayoutGrid2D,
AecLayoutGrid3D, AecLayoutTool, AecSlice,
AecVolumeLayoutTool
The object this method applies to.

Point

Variant (three-element array of doubles); input-only
A 3D point near a layout node.

RetVal

Long
The index of the closest node.

Remarks

The node index is also returned by the Node property of
AecAnchorEntToLayoutNode.
Contains Method

Tests to see if a ring is inside another ring.

See Also | Example

Signature

RetVal = object.Contains(Ring)

object

AecRing The object this method applies to.

Ring

AecRing; input-only
The AecRing to check.

RetVal

Boolean
TRUE: The ring is inside.
FALSE: The ring is not inside.
ConvertToCurrentAreaDisplay Method

Converts an area value using the LinearUnit and AreaDisplayUnit settings.

See Also | Example

Signature

RetVal = object.ConvertToCurrentAreaDisplay(Area)

object

(DatabasePreferences) The object this method 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 this method applies to.

Volume

Double; input-only
The volume to convert.

RetVal

Double
The resulting volume.
Copy Method

Duplicates the given object to the same location.

See Also | Example

Signature

RetVal = object.Copy

Object


The object to be copied.

RetVal

Object

The newly created duplicate object.

Remarks

NOTE You cannot execute this method while simultaneously iterating through a collection. An iteration will open the work space for a read-only operation, while this method attempts to perform a read-write operation. Complete any iteration before you call this method.
CopyFrom Method

Copies AecRings and AecProfiles.

See Also | Example

Signature

object.CopyFrom SourceObject

object

AecProfile, AecRing
The object being copied to.

SourceObject

The source object to be copied; input only

When copying a Profile, the source object must be an AecProfile object.

When copying a Ring, the source object must be an AecRing object.
CopyObjects Method

Duplicates multiple objects (deep cloning).

See Also | Example

Signature

RetVal = object.CopyObjects(Objects[, Owner][, IDPairs])

Object

AecBaseDatabase, AecDatabase
The object or objects this method applies to.

Objects

Variant (array of objects); input-only
The array of primary objects to be copied. All the objects must have the same owner, and the owner must belong to the database or document that is calling this method.

Owner

Variant (a single object); input-only; optional
The new owner for the copied objects. If no owner is specified, the objects will be created with the same owner as the objects in the Objects array.

IDPairs

Variant (array of IDPair objects); input-output; optional
Information on what happened during the copy and translation process.
Input: an empty variant.
Output: an array of IDPair objects.

RetVal

Variant (array of objects)
An array of newly created duplicate objects. Only
primary objects are returned in this array. For more information on what occurred during the CopyObjects operation, or a list of objects owned by primary objects that were also copied, consult the IDPairs array.

Remarks

To copy objects to another open drawing, set the Owner parameter to the other drawing's model space.

During the CopyObjects operation, objects that are owned or referenced by the primary objects in the Objects parameter will also be copied.

NOTE You cannot execute this method while simultaneously iterating through a collection. An iteration will open the workspace for a read-only operation, while this method attempts to perform a read-write operation. Complete any iteration before you call this method.
Delete Method

Deletes the specified object.

See Also | Example

Signature

object.Delete

object

    The object to be deleted.
**ExpandKey Method**

Calculates and returns the Layer Name that would be generated given the supplied 'Key'.

See Also | [Example](#)

**Signature**

RetVal = object.ExpandKey(key)

object

AecLayerKeyStyle The object this method applies to.

key

String; input-only
The key to convert.

RetVal

String; Layer Name

**Remarks**

If the Key is not valid, an empty string is returned.
ExportFreeForm Method

Exports a free form representation to a file.

Signatures: Overview

Mass Elements

object.ExportFreeForm("FileName", CenterAtOrigin)

Mass Groups

object.ExportFreeForm("FileName")

Signatures: Detail

object.ExportFreeForm("FileName", CenterAtOrigin)

object

AecMassElement The object being exported.

FileName

String; input-only
The name of the file to export the free form representation to.

CenterAtOrigin

Boolean; optional
Determines whether the original coordinates of the boundary representation are stored, or the coordinates are translated to WCS.
TRUE: Use the original coordinates of the boundary representation.
FALSE: Translate the vertices of the boundary representation to WCS.

The default is TRUE.

object.ExportFreeForm("FileName")

object
    AecMassGroup
    The object being exported.

FileName
    String; input-only
    The name of the file to export the free form representation to.

Remarks

If no path is specified for FileName, the file is created in the current drawing's directory. If you do not specify a file extension, none is added.
FromPoints Method

Makes a ring from 2D Points.

See Also | Example

**Signature**

object.FromPoints(points)

object  
AecRing The object this method applies to.

points  
Variant(n-element array of doubles); input-only  
List of 2D points.

**Remarks**

The input for this method takes a variant array of doubles representing 2D points.
FromPolyline Method

Makes a ring from a 2D Polyline.

Signature

object.FromPolyline(polyline)

object

AecRing The object this method applies to.

polyline

AcadLWPolyline; input-only
2D polyline to copy the graphics from.
GenerateLayer Method

Generate the layer record, given a Layer Key.

See Also | Example

Signature

RetVal = object.GenerateLayer(Key)

object

AecLayerKeyStyle The object this method applies to.

Key

String; input-only

RetVal

AcadLayer
The AcadLayer will be created if necessary, using the settings from the Layer Standard for Name, Color, Linetype, etc.
GetAnchor Method

Returns the anchor attached to an object.

See Also | Example

Signature

RetVal = object.GetAnchor()

object

AecBlockRef, AecCamera, AecCellLayoutTool, AecClipVol, AecClipVolRes, AecEntRef, AecGeo, AecGridAssembly, AecLayoutCurve, AecLayoutGrid2D, AecLayoutGrid3D, AecLayoutTool, AecMaskBlockRef, AecMassElement, AecMassGroup, AecMVBlockRef, AecPolygon, AecSlice, AecVolumeLayoutTool

The object the anchor is attached to.

RetVal

AecAnchor, AecAnchorEntToCurve, AecAnchorEntToGridAssembly, AecAnchorEntToLayoutCell, AecAnchorEntToLayoutNode, AecAnchorEntToLayoutVolume, AecAnchorLeadEntToNode, AecAnchorToRef

The anchor attached to the object.

Remarks
Use an AecAnchor object as a generic return argument. (The example code for this method demonstrates how to access an anchor attached to any AEC object.)
GetBoundingBox Method

Gets two points of a box enclosing the specified object.

See Also | Example

Signature

object.GetBoundingBox MinPoint, MaxPoint

Object


MinPoint

Variant (three-element array of doubles); output-only
The 3D WCS coordinates specifying the minimum point of the object's bounding box.

MaxPoint

Variant (three-element array of doubles); output-only
The 3D WCS coordinates specifying the maximum point of the object's bounding box.

Remarks

The corners are returned in WCS coordinates with the box edges parallel to the WCS X, Y, and Z axes.

MaxPoint
MinPoint
GetExtensionDictionary Method

Gets the extension dictionary associated with an object.

See Also | Example

Signature

RetVal = object.GetExtensionDictionary

Object

AecAnchor, AecAnchorEntToCurve, AecAnchorEntToGridAssembly, AecAnchorEntToLayoutCell, AecAnchorEntToLayoutNode, AecAnchorEntToLayoutVolume, AecAnchorExtendedTagToEnt, AecAnchorLeadEntToNode, AecAnchorToRef, AecBlockRef, AecCamera, AecCellLayoutTool, AecClipVol, AecClipVolRes, AecDictionary, AecDictRecord, AecEntity, AecEntRef, AecGeo, AecGridAssembly, AecLayerKeyStyle, AecLayerKeyStyles, AecLayoutCurve, AecLayoutGrid2D, AecLayoutGrid3D, AecLayoutTool, AecMaskBlockRef, AecMaskBlockStyle, AecMaskBlockStyles, AecMassElement, AecMassGroup, AecMVBlockRef, AecMVBlockStyle, AecMVBlockStyles, AecObject, AecPolygon, AecPolygonStyle, AecPolygonStyles, AecProfileStyle, AecProfileStyles, AecSlice, AecVolumeLayoutTool

The object or objects this method applies to.

RetVal

Dictionary; output-only
The extension dictionary for the object.
Remarks

If an object does not have an extension dictionary, this method will create a new extension dictionary for that object and return it in the return value.

You can query an object to see if it has an extension dictionary by using the `HasExtensionDictionary` property.
GetXData Method

Gets the extended data (XData) associated with an object.

See Also | Example

Signature

object.GetXData AppName, XDataType, XDataValue

Object

AecAnchor, AecAnchorEntToCurve,
AecAnchorEntToGridAssembly,
AecAnchorEntToLayoutCell,
AecAnchorEntToLayoutNode,
AecAnchorEntToLayoutVolume,
AecAnchorLeadEntToNode,
AecAnchorExtendedTagToEnt AecAnchorToRef,
AecBlockRef, AecCamera, AecCellLayoutTool,
AecClipVol, AecClipVolRes, AecDictionary,
AecDictRecord, AecEntity, AecEntRef, AecGeo,
AecGridAssembly, AecLayerKeyStyle,
AecLayerKeyStyles, AecLayoutCurve,
AecLayoutGrid2D, AecLayoutGrid3D,
AecLayoutTool, AecMaskBlockRef,
AecMaskBlockStyle, AecMaskBlockStyles,
AecMassElement, AecMassGroup,
AecMVBlockRef, AecMVBlockStyle,
AecMVBlockStyles, AecObject, AecPolygon,
AecPolygonStyle, AecPolygonStyles,
AecProfileStyle, AecProfileStyles, AecSlice,
AecVolumeLayoutTool

The object or objects this method applies to.

AppName

String; input-only
A NULL string will return all the data attached to the object, regardless of the application that created it.
Supplying an application name will return only the data that was created by the specified application.

**XDataType**

Variant (array of shorts); output-only

**XDataValue**

Variant (array of variants); output-only

**Remarks**

Extended data can be used to store instance-specific data created by custom applications. This data can be added to any object. This data follows the object's definition data, and is maintained in the order that it was saved into the document. (AutoCAD maintains this information, but doesn't use it.)
HandleToObject Method

Gets the object that corresponds to the given handle.

See Also | Example

Signature

RetVal = object.HandleToObject(Handle)

Object

AecBaseDatabase, AecDatabase
The object or objects this method applies to.

Handle

String; input-only
The handle of the object to return.

RetVal

Object
The object that corresponds to the given handle.

Remarks

This method can only return objects in the current document.
Has Method

Determines if an object is a member of the specified collection.

See Also | Example

Signature

RetVal = object.Has("Name")

object

AecLayerKeyStyles, AecMaskBlockStyles, AecMVBBlockStyles, AecPolygonStyles, AecProfileStyles, AecDisplayRepresentations
The collection this method applies to.

Name

String; input-only
The name of the object to check for.

RetVal

Boolean
TRUE: Collection contains the object.
FALSE: Collection does not contain the object.
Highlight Method

Sets the highlight status for the given object, or for all objects in a given selection set.

See Also | Example

Signature

object.Highlight HighlightFlag

Object


The object or objects this method applies to.

HighlightFlag

Boolean; input-only

TRUE: The object is highlighted.

FALSE: The existing highlight is removed from the object.

Remarks

Once the highlight flag for an object has been set, a call to the Update or Regen method is required to view the change.

Note that this function does not return the current highlight status of an object.
ImportFreeForm Method

Imports a free form representation from a file.

**Signature**

```csharp
object.ImportFreeForm("FileName", [CenterAtLocation])
```

- **object**
  - `AecMassElement` The object being imported to. This must be a free form mass element (`Type` = `aecMassElementTypeFreeForm`).

- **FileName**
  - String; input-only
  - The name of the file to import the free form representation from.

- **CenterAtLocation**
  - Boolean; optional.
  - Controls whether the the vertices of the imported boundary representation are translated to the insertion point of the mass element.
  - **TRUE**: Translate the boundary representation to the mass element insertion point.
  - **FALSE**: Use the coordinates saved with the exported boundary representation.

  The default is **FALSE**.

See Also | Example
Remarks

If no path is specified for *FileName*, the current drawing's directory is searched.
Init Method

Initialization function which must be called before the AEC database object can be accessed

See Also | Example

Signature

object.Init(AcadDatabaseObject)

object

AecBaseDatabase, AecBaseDatabasePreferences
The object this method applies to.

AcadDatabaseObject

AcadDatabase; input-only
The database.
Intersect Method

Boolean Intersects an AecProfile with the current AecProfile.

See Also | Example

Signature

object.Intersect(Profile)

object

AecProfile The object this method applies to.

Profile

AecProfile; input-only
The profile to intersect with the current profile.
**IntersectWith Method**

Gets the points where one object intersects another object in the drawing.

See Also | [Example](#)

**Signature**

RetVal = object.IntersectWith(IntersectObject, ExtendOption)

- **Object**
  - The object or objects this method applies to.

- **IntersectObject**
  - Object, input-only;
  - This can be any drawing entity object.

- **ExtendOption**
  - AcExtendOption enum; input-only
  - This option specifies if one or the other, both, or none of the entities are to be extended in order to attempt an intersection.

  - acExtendNone
    - Does not extend either object.

  - acExtendThisEntity
    - Extends the base object.
acExtendOtherEntity    Extends the object passed as an argument.

acExtendBoth    Extends both objects.

RetVal

Variant (array of doubles)
The array of points where one object intersects another object in the drawing.

Remarks

If the two objects do not intersect, no data is returned. You can request the point of intersection that would occur if one or both of the objects were extended to meet the other. For example, suppose a drawing contains the three lines shown in the illustration below. Line1 is the base object from which this method was called and line3 is the object passed as a parameter. If the ExtendOption passed is acExtendThisEntity, point A is returned as the point where line1 would intersect line3 if line1 were extended. If the ExtendOption is acExtendOtherEntity, no data is returned because even if line3 were extended, it would not intersect line1.

If the intersection type is acExtendBothEntities and line2 is passed as the parameter entity, point B is returned. If the ExtendOption is acExtendNone and line2 is the parameter entity, no data is returned.

\[
\begin{align*}
\text{line3} \\
\text{line2} \\
\text{line1}
\end{align*}
\]
**Item Method**

Returns a member object of a collection.

See Also | [Example](#)

**Signature**

RetVal = object.Item(Index)

- **object**
  - All Collections, AecDisplayConfigurations, AecDisplaySets, AecDisplayRepresentations, AecDisplayComponents, AecMVBlockDisplayRepDefs, AecViewBlockDefs, AecsConnectionNodes
  - The collection this method 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.

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

**Remarks**
This method supports case-sensitive string based iteration. For example, you could reference the Multi-View Block definition named "Fixture1" through the following statement:

    Set mvBlock = cMvBlocks.Item("Fixture1")
Mirror Method

Creates a mirror image copy of a planar object around an axis.

See Also | Example

Signature

RetVal = object.Mirror(Point1, Point2)

Object


The object or objects this method applies to.

Point1

Variant (three-element array of doubles); input-only
The 3D WCS coordinates specifying the first point of the mirror axis.

Point2

Variant (three-element array of doubles); input-only
The 3D WCS coordinates specifying the second point of the mirror axis.

RetVal

Mirrored object
This can be any drawing entity object.

Remarks
The two points specified as parameters become the endpoints of a line around which the base object is reflected.

*Selected object*

*Point1 and Point2 specifying the mirror axis*

*Mirrored object*

This method places the reflected image into the drawing and retains the original object. To remove the original object, use the Delete method.

You can mirror a Viewport object in paper space, although doing so has no affect on its model space view or on model space objects.

AutoCAD checks to see if the object to be copied owns any other object. If it does, it performs a copy on those objects as well. The process continues until all owned objects have been copied.

NOTE You cannot execute this method while simultaneously iterating through a collection. An iteration will open the work space for a read-only operation, while this method attempts to perform a read-write operation. Complete any iteration before you call this method.
**Mirror3D Method**

Creates a mirror image of the given object about a plane.

See Also | Example

**Signature**

RetVal = object.Mirror3D(Point1, Point2, Point3)

**Object**


The object or objects this method applies to.

**Point1**

Variant (three-element array of doubles); input-only
The 3D WCS coordinates specifying the first point of the mirror plane.

**Point2**

Variant (three-element array of doubles); input-only
The 3D WCS coordinates specifying the second point of the mirror plane.

**Point3**

Variant (three-element array of doubles); input-only
The 3D WCS coordinates specifying the third point of the mirror plane.

**RetVal**

Mirrored object
This object can be any drawing entity object.

Remarks

Object mirrored about a plane defined by three points

AutoCAD checks to see if the object to be copied owns any other object. If it does, it performs a copy on those objects as well. The process continues until all owned objects have been copied.

NOTE You cannot execute this method while simultaneously iterating through a collection. An iteration will open the work space for a read-only operation, while this method attempts to perform a read-write operation. Complete any iteration before you call this method.
**Move Method**

Moves an object along a vector.

See Also | **Example**

**Signature**

object.Move Point1, Point2

**Object**


The object or objects this method applies to.

**Point1**

Variant (three-element array of doubles); input-only

The 3D WCS coordinates specifying the first point of the move vector.

**Point2**

Variant (three-element array of doubles); input-only

The 3D WCS coordinates specifying the second point of the move vector.

**Remarks**

The two points you specify define a displacement vector indicating how far the given object is to be moved and in what direction.
Given object with two points indicated  Moved object
**NodeLocation Method**

Returns the location of the node at a given index.

**Signature**

RetVal = object.NodeLocation(Node)

**Object**

- AecCellLayoutTool, AecClipVol, AecGridAssembly,
- AecLayoutCurve, AecLayoutGrid2D,
- AecLayoutGrid3D, AecLayoutTool, AecSlice,
- AecVolumeLayoutTool

The object this method applies to.

**Node**

Long; input-only
Index of the node.

**RetVal**

Variant (three-element array of doubles)
The location of the node.

**Remarks**

The node index is returned by the Node property of AecAnchorEntToLayoutNode.
**ObjectIdToObject Method**

Gets the object that corresponds to the given object ID.

See Also | [Example](#)

**Signature**

RetVal = object.ObjectIdToObject(ID)

**Object**

AecBaseDatabase, AecDatabase
The object or objects this method applies to.

**ID**

Long; input-only
The object ID of the object to return.

**RetVal**

Object
The object that corresponds to the given object ID.

**Remarks**

This method can only return objects in the current document.
ReleaseAnchor Method

Removes an anchor from an object.

See Also | Example

Signature

object.ReleaseAnchor()

object


The object this method applies to.
Remove Method

Removes an item from a collection.

Signatures: Overview

AecDictionary,
Layer Key Styles,
Mask Block Styles,
Mass Element Styles,
Multi-view Block Styles,
Polygon Styles,
Profile Styles

collection.Remove(Name)

Layer Keys,
Rings

collection.Remove(Index)

ViewBlockDefs

collection.Remove(AcadBlock)

Signatures: Detail

object.Remove (Name)

object

AecDictionary, AecLayerKeyStyles,
AecMaskBlockStyles, AecMassElementStyles,
AecMVBlockStyles, AecPolygonStyles,
The collection this method applies to.

Name

String; input-only
The name of the item to be removed.

Index

Long; input-only
The index number value of the item to remove from the collection.

The collection you are removing a block from.

The block to remove from the collection.
**Rename Method**

Renames an item in an AecDictionary.

**Signature**

RetVal = object.Rename(OrigName, NewName)

**Object**

- AecDictionary, AecLayerKeyStyles,
  AecMaskBlockStyles, AecMassElementStyles,
  AecMVBlockStyles, AecPolygonStyles,
  AecProfileStyles
  The object this method applies to.

**OrigName**

String; input-only
The original name of the item.

**NewName**

String; input-only
The new name of the item.

**Remarks**

Objects will maintain their reference to a style even if it has been renamed.
**Rotate Method**

Rotates an object around a base point.

See Also | Example

**Signature**

object.Rotate BasePoint, RotationAngle

Object


AecPolygon, AecSlice, AecVolumeLayoutTool

The object or objects this method applies to.

BasePoint

Variant (three-element array of doubles); input-only

The 3D WCS coordinates specifying the point through which the axis of rotation is defined as parallel to the Z axis of the UCS.

RotationAngle

Double; input-only

The angle in radians to rotate the object. This angle determines how far an object rotates around the base point relative to its current location.

**Remarks**
**Rotate3D Method**

Rotates an object around a 3D axis. Point1 and Point2 define the line that becomes the axis of rotation.

See Also | Example

**Signature**

object.Rotate3D Point1, Point2, RotationAngle

**Object**


The object or objects this method applies to.

**Point1**

Variant (three-element array of doubles); input-only
The 3D WCS coordinates specifying the first point of the axis line.

**Point2**

Variant (three-element array of doubles); input-only
The 3D WCS coordinates specifying the second point of the axis line.

**RotationAngle**

Double; input-only
The angle in radians to rotate the object about the selected axis.
**SaveAsDefault Method**

Saves the DatabasePreferences to the registry.

See Also | Example

**Signature**

object.SaveAsDefault()

object  

*AecBaseDatabasePreferences* The object this method applies to.
ScaleEntity Method

Scales an object equally in the X, Y, and Z directions.

See Also | Example

Signature

object.ScaleEntity BasePoint, ScaleFactor

Object

- AecBlockRef
- AecCamera
- AecCellLayoutTool
- AecClipVol
- AecClipVolRes
- AecEntity
- AecEntRef
- AecGeo
- AecGridAssembly
- AecLayoutCurve
- AecLayoutGrid2D
- AecLayoutGrid3D
- AecLayoutTool
- AecMaskBlockRef
- AecMassElement
- AecMassGroup
- AecMVBBlockRef
- AecPolygon
- AecSlice
- AecVolumeLayoutTool

The object or objects this method applies to.

BasePoint

Variant (three-element array of doubles); input-only
The 3D WCS coordinates specifying the base point.

ScaleFactor

Double; input-only
The factor by which to scale the object. The dimensions of the object are multiplied by the scale factor. A scale factor greater than 1 enlarges the object. A scale factor between 0 and 1 reduces the object. The scale factor must be greater than 0.0.

Remarks
BasePoint

ScaleFactor = 0.5 and ScaleFactor = 2
**ScaleXY Method**

Scales a profile in the X and Y directions.

See Also | Example

**Signature**

object.ScaleXY (X, Y)

object

*AecProfile* The object this method applies to.

**X**

Double; input-only
The X factor to scale by.

**Y**

Double; input-only
The Y factor to scale by.
SetXData Method

Sets the extended data (XData) associated with an object.

See Also | Example

Signature

object.SetXData XDataType, XData

Object

AecAnchor, AecAnchorEntToCurve,
AecAnchorEntToGridAssembly,
AecAnchorEntToLayoutCell,
AecAnchorEntToLayoutNode,
AecAnchorEntToLayoutVolume,
AecAnchorExtendedTagToEnt
AecAnchorLeadEntToNode, AecAnchorToRef,
AecBlockRef, AecCamera, AecCellLayoutTool,
AecClipVol, AecClipVolRes, AecDictionary,
AecDictRecord, AecEntity, AecEntRef, AecGeo,
AecGridAssembly, AecLayerKeyStyle,
AecLayerKeyStyles, AecLayoutCurve,
AecLayoutGrid2D, AecLayoutGrid3D,
AecLayoutTool, AecMaskBlockRef,
AecMaskBlockStyle, AecMaskBlockStyles,
AecMassElement, AecMassGroup,
AecMVBlockRef, AecMVBlockStyle,
AecMVBlockStyles, AecObject, AecPolygon,
AecPolygonStyle, AecPolygonStyles,
AecProfileStyle, AecProfileStyles, AecSlice,
AecVolumeLayoutTool

The object or objects this method applies to.

XDataType

Variant (array of short); input-only
See Extended Data in the *DXF Reference* for the extended data group codes to specify for this argument.

**XData**

Array of Variant; input-only

**Remarks**

Extended data can be used for storing instance-specific data created by custom applications. This data can be added to any entity. This data follows the entity's definition data and is maintained in the order in which it was saved into the document. (AutoCAD maintains this information but doesn't use it.)
Subtract Method

Subtracts an AecProfile from the current AecProfile.

See Also | Example

Signature

object.Subtract(Profile)

object

AecProfile The object this method applies to.

Profile

AecProfile; input-only
The profile to subtract.
## TransformBy Method

Moves, scales, or rotates an object given a 4x4 transformation matrix.

### Signature

`object.TransformBy TransformationMatrix`

**Object**


The object or objects this method applies to.

**TransformationMatrix**

Variant (4x4 array of doubles); input-only

A 4x4 matrix specifying the transformation to perform.

### Remarks

The following table demonstrates the transformation matrix configuration, where $R =$ Rotation, and $T =$ Translation:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>R00</td>
<td>R01</td>
<td>R02</td>
<td>T0</td>
</tr>
<tr>
<td>R10</td>
<td>R11</td>
<td>R12</td>
<td>T1</td>
</tr>
<tr>
<td>R20</td>
<td>R21</td>
<td>R22</td>
<td>T2</td>
</tr>
</tbody>
</table>
This method will return an error if the transformation matrix is not correct.

Sample transformation matrices are provided in the example code for this method.
Update Method

Updates the object in the drawing window.

See Also | Example

Signature

object.Update

Object


The object or objects this method applies to.
Properties

**AdjustSizing** Specifies if the size of the anchored object is adjusted to match the size of the grid cell.

**AllowLongSymbolNames** Determines if symbol names may include extended character sets, or more than 31 characters.

**AllowVariation** Specifies if the anchored object is allowed to vary.

**AlternateName** The AlternateName of the Style.

**AlwaysImportLayerStandard** Specifies whether to always import a newer layer standard.

**Angle** Specifies the angle of the layout grid.

**Angle1** Deviation of cut plane at starting edge where zero is perpendicular to the clip line.

**Angle2** Deviation of cut plane at ending edge where zero is perpendicular to the clip Line.

**AngleFromNode** Specifies the angle from the anchored entity to the node.

**AngularAzimuth** The format used to display angles.

**AngularDisplayFormat** Specifies the unit used when angular measurements are displayed.

**AngularPrecision** Specifies the angular precision for the drawing.
**Application** Gets the Application object.

**ApplyResize** Specifies if the anchored object is resized when the layout cell is resized.

**Area** Gets the area of an object.

**AreaDisplayUnit** The unit used when area measurements are displayed.

**AreaPrecision** The precision to which area measurements are displayed.

**AreaSuffix** The suffix displayed after an area unit in the drawing.

**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.

**BayAngle** Specifies the bay angle of the layout grid.

**BlockBasedLayerOffBehavior** Immediate and independent layer on/off control of display components.

**Blocks** Gets the Blocks collection for the drawing.

**BottomOffset** Specifies the bottom offset of the anchored object to the grid it is attached to.

**Boundary** Specifies the boundary object attached to the grid.

**Bounds** Gets the upper left and lower right corners of the profile.

**C**

**Cell** Specifies the grid cell to anchor the entity to.

**CenterOnNode** Specifies if the anchored object is centered on the layout node.

**Centroid** Gets the center of area or mass for a profile shape.
**Color** Specifies the object's color.

**ContourlinesPerSurface** Specifies the number of contour lines (isolines) per surface on objects.

**CoordinatePrecision** Specifies the precision to which coordinate measurements are displayed.

**Coordinates** Returns the vertices of a ring or profile.

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

**CreateDimscaleOverride** Automatically create a dimscale override.

**Database** Gets the database in which the object belongs.

**DatabaseScale** Specifies the database scale at which the horizontal axis of the drawing is displayed.

**Depth** Gets or sets the size of the grid or mass element in its relative Y direction.

**Description** Gets or sets text describing the AEC Object.

**Dictionaries** Gets the Dictionaries collection for the document.

**DimStyles** Gets the DimStyles collection for the document.

**DisplaySilhouette** Controls if silhouette curves of solid objects are displayed in Wireframe mode.

**Document** Gets the document (drawing) in which the object belongs.

**EdgeWidth** Specifies the edge width of the polygon style.

**EdgeWidthJustify** Specifies whether edges are displayed inside, outside, or centered on the AEC Polygon vertices.
**ElevationModelSpace** Specifies the elevation setting in the model space.

**ElevationPaperSpace** Specifies the elevation setting in the paper space.

**ElevationPrecision** Specifies the precision to which elevation measurements are displayed in the drawing.

**EndOffset** Specifies the offset distance from the end point of the curve, which may be used to reduce the effective layout curve length.

**FacetDeviation** Specifies the allowed deviation when facetting curved lines and surfaces.

**FacetMaximum** Specifies the maximum number of facets used for a circle.

**FlipX** Determines if the anchored object is flipped about the X axis of the curve it is anchored to.

**FlipY** Determines if the anchored object is flipped about the Y axis of the curve it is anchored to.

**FlipZ** Determines if the anchored object is flipped about the Z axis of the curve it is anchored to.

**Groups** Gets the Groups collection for the document.

**Handle** Gets the handle of an object.

**HasExtensionDictionary** Determines if the object has an extension dictionary associated with it.

**Height** Specifies the height of the object.

**Hyperlinks** Gets the Hyperlinks collection for an entity.
Index Returns the position of the item within its collection.

InsertionPoint Specifies the insertion point of the profile.

InsideRadius Specifies the inside radius of the layout grid.

Keys Returns the collection of layer keys in a layer key style.

Layer Specifies the layer the object is on.

LayerFile Specifies the name of the file containing default layer standard and key style definitions.

LayerKeyStyle Specifies the layer key style for the object.

LayerKeyStyles Returns the collection of layer key styles in the current drawing.

Layers Gets the Layers collection for the document.

LayerStandard The layer standard to apply to the drawing.

LayoutCurve Provides access to layout curve geometry.

LayoutGrid2D Identifies the 2D layout grid that owns the object.

LayoutGrid3D Identifies the 3D layout grid that owns the object.

Layouts Gets the Layouts collection for the document.

LeaderExtension1 Specifies the first leader extension.

LeaderExtension2 Specifies the second leader extension.
**LeftOffset** Specifies the left offset for the anchored object.

**Limits** Specifies the drawing limits.

**LinearDisplayFormat** Specifies the format used to display linear measurement values.

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

**LinearUnit** Specifies the unit used to display linear values.

**LineType** Specifies the linetype of the object.

**Linetypes** Gets the Linetypes collection for the document.

**LinetypeScale** Specifies the linetype scale of an entity.

**LineWeight** Specifies the lineweight of the object.

**LineweightDisplay** Specifies whether lineweights are displayed in model space for the drawing.

**Location** Specifies the insertion position of the object.

**LowerExtension** Specifies the distance of the lower cut plane below the clip volume position.

**M**

**MaskBlockStyles** Returns the collection of mask block styles in the specified drawing document or database.

**MassElementStyles** Returns the collection of mass element styles in the specified drawing database.

**MassGroup** Gets or sets the parent mass group of a mass element or mass group.

**MassGroupName** Specifies the mass group that the mass element or group is currently attached to.
MassGroups Returns the collection of mass groups in a document or database.

MaxActiveViewports Specifies the maximum number of active viewports.

MeasurementUnit Identifies the system of units used to display measurements in the drawing.

ModelSpace Gets the ModelSpace collection for the document.

MVBlockRef Gets or sets a multi-view block reference for a view block or collection of view blocks.

MVBlockStyles Returns the collection of multi-view block styles in the specified document or database.

N

Name Returns the name of the object.

Node Specifies the layout node the anchor attaches an object to.

Nodes Returns the collection of nodes on the layout curve.

Normal Specifies the normal vector of the AEC object.

NorthRotation Specifies the angle aligned to north within a drawing.

O

ObjectId Gets the object ID of the object.

ObjectName Gets the AutoCAD class name of the object.

ObjectSortByPlotting Toggles sorting of drawing objects by plotting order.

ObjectSortByPSOutput Toggles sorting of drawing objects by PostScript output order.

ObjectSortByRedraws Toggles sorting of drawing objects by redraw order.
**ObjectSortByRegens** Toggles sorting of drawing objects by regeneration order.

**ObjectSortBySelection** Toggles sorting of drawing objects by object selection.

**ObjectSortBySnap** Toggles sorting of drawing objects by object snap.

**Offset** Specifies the offset from the layout node.

**OLELaunch** Determines whether to launch the parent application when plotting OLE objects.

**Operation** Specifies how objects are combined.

**OverridesEnabled** Specifies whether layer overrides are applied to the style when a layer is generated from a key.

**OverrideSettings** Returns the collection of layer key override settings for a layer key style.

**OwnerID** Gets the object ID of the owner (parent) object.

**PaperSpace** Gets the PaperSpace collection for the document.

**PlotConfigurations** Gets the PlotConfigurations collection for the document.

**PlotStyleName** Specifies the plotstyle name for the object.

**Plottable** Specifies whether the layer generated from the key is plottable.

**PolygonStyles** Returns the collection of polygon styles in the specified drawing database.

**Preferences** Returns the Preferences object for the specified drawing.

**Profile** Returns or sets the object’s profile.
ProfileStyle  Specifies the profile style of the mass element.

ProfileStyleName  Specifies the name of the mass element's profile style.

ProfileStyles  Returns the collection of profiles in the specified drawing database.

ProjectName  Specifies the name of the project that the drawing belongs to.

Q

R

Radius  Specifies the radius of curved mass elements.

Reference  Specifies the object being anchored to.

RegisteredApplications  Gets the RegisteredApplications collection for the document.

Removeable  Specifies whether you can remove the layer key from the layer key style.

RenderSmoothness  Specifies the smoothness of shaded, rendered, and hidden line-removed objects.

ResizeOffset  Specifies the offset of the anchored object to the cell boundary when the cell is resized.

RightOffset  Specifies the right offset for the anchored object.

Rings  Returns the collection of rings that make up the profile.

Rise  Specifies the rise of gabled mass elements.

Rotation  Specifies the rotation reference angle for the object.

S

ScaleOnInsert  Specifies whether objects are automatically scaled when
inserted into the drawing.

**ScaleX** Specifies the X direction scale factor of the inserted block or entity.

**ScaleY** Specifies the Y direction scale factor of the inserted block or entity.

**ScaleZ** Specifies the Z direction scale factor of the inserted block or entity.

**SegmentPerPolyline** Specifies the number of line segments to be generated for each polyline curve.

**SelfIntersects** Determines if the profile is self-intersecting.

**Shape** Specifies the shape of the layout grid.

**Side1** Specifies the length of the cut plane along the starting edge.

**Side2** Specifies the length of the cut plane along the ending edge.

**SolidFill** Specifies if multilines, traces, solids, all hatches (including solid-fill) and wide polylines are filled in.

**Spacing** Specifies the spacing between nodes or for an individual node on a layout tool.

**StartOffset** Specifies the offset distance from the start point of the curve, which may be used to reduce the effective layout curve length.

**Style** Specifies the style of the object.

**StyleName** Specifies the name of the style that applies to the object.

**T**

**TextFrameDisplay** Toggles the display of frames for text objects instead of displaying the text itself.

**TextHeight** Specifies the default height of text in the drawing.
**TextStyles** Gets the TextStyles collection for the document.

**TopOffset** Specify the top offset for the anchored object.

**TrueColor** Returns the true color of the object.

**Type** For layout curves, specifies how nodes are specified along the curve; for mass elements, specifies the shape of the mass element.

**UseModelExtents** Determines if the height and lower extension are set automatically using model extents.

**UseNodeCS** Specifies whether to use the coordinate system of the layout node.

**UserCoordinateSystems** Gets the UCSs collection for the document.

**Valid** Tests the validity of the object.

**Value** For layer overrides, specifies the value of the layer key style override; for layout curves, specifies either the spacing between nodes or the number of nodes on the layout curve.

**VerticalScale** Specifies the scale at which the vertical axis of the drawing is displayed (in Land Development Desktop only).

**ViewBlocks** Returns the collection of view blocks used for display representations by the multi-view block reference.

**Viewports** Gets the Viewports collection for the document.

**Views** Gets the Views collection for the document.

**Visible** Specifies whether the object or application is visible.

**Void** Specifies if a ring produces a void in a profile.
**Volume** Returns the volume of the mass element or mass group.

**VolumeDisplayUnit** Specifies the units used to display volume measurements.

**VolumePrecision** Specifies the number of decimal places displayed for volume measurements.

**VolumeSuffix** The suffix string added when volume measurements are displayed in the drawing.

**W**

**Width** Specifies the width of the object.

**WindowState** Specifies the state of the application or document window.

**WindowTitle** Gets the title of the document window.

**X**

**XCount** Specifies the number of nodes along the X direction.

**XDistance** Specifies the distance along the curve's X axis to the anchored object.

**XEndOffset** Specifies the offset distance from the end point along the X direction grid curve, which may be used to reduce the grid size.

**XNodes** Returns the collection of nodes along the X direction of the layout grid.

**XOffset** Specifies the offset in the X direction of the view block within the multi-view block.

**XPositionFrom** Specifies the position on the curve from which the XDistance to an anchored object is measured.

**XPositionTo** Specifies the position on the anchored object to measure XDistance to.
**XRefEdit** Determines if the current drawing can be edited in place when being referenced by another user.

**XRefLayerVisibility** Determines the visibility of xref-dependent layers and specifies if nested xref path changes are saved.

**XrefOverlaysUseOwnDisplayConfig** When used as an Xref Overlay, use this drawing's default display configuration initially.

**XRotation** Specifies the rotation of the anchored object around the X axis.

**XSpacing** Specifies the spacing between nodes along the X direction of the layout grid.

**XStartOffset** Specifies the offset distance from the start point along the X direction grid curve, which may be used to reduce the grid size.

**XType** Specifies the rule controlling node spacing along the X direction.

**Y**

**YAlignment** Specifies the alignment of the anchored object in relation to the baseline of the assembly.

**YCount** Specifies the number of nodes along the y direction.

**YDistance** Specifies the distance along the curve's Y axis to the anchored object.

**YEndOffset** Specifies the offset distance from the end point along the Y direction grid curve, which may be used to reduce the grid size.

**YNodes** Returns the collection of nodes along the Y direction of the layout grid.

**YOffset** Specifies the Y offset of the anchored object, or the offset in the Y direction of the view block within the multi-view block.

**YPositionFrom** Specifies the position on the curve from which the YDistance to an anchored object is measured.
**YPositionTo** Specifies the position on the anchored object to measure YDistance to.

**YRotation** Specifies the rotation of the anchored object around the layout node’s Y axis.

**YSpacing** Specifies the spacing between nodes along the Y direction of the layout grid.

**YStartOffset** Specifies the offset distance from the start point along the Y direction grid curve, which may be used to reduce the grid size.

**YType** Specifies the rule controlling node spacing along the Y direction.

**Z**

**ZCount** Specifies the number of nodes along the Z direction.

**ZDistance** Specifies the distance above the curve to the anchored object.

**ZEndOffset** Specifies the offset distance from the end point along the Z direction grid curve, which may be used to reduce the grid size.

**ZNodes** Returns the collection of nodes along the Z direction of the 3D layout grid.

**ZOffset** Specifies the offset in the Z direction of the view block within the multi-view block.

**Zoom** Specifies the focal length of the camera, in millimeters.

**ZPositionFrom** Specifies the position on the curve from which the ZDistance to an anchored object is measured.

**ZPositionTo** Specifies the position on the anchored object to measure ZDistance to.

**ZRotation** Specifies the rotation of the anchored object around the Z axis.

**ZSpacing** Specifies the spacing between nodes along the Z direction of the layout grid.
ZStartOffset  Specifies the offset distance from the start point along the Z direction grid curve, which may be used to reduce the grid size.

ZType  Specifies the rule controlling node spacing along the Z direction.
**ActiveSet Property**

Returns the active display set for the specified view port.

See Also | Example

**Signature**

RetVal = object.ActiveSet(ViewPort)

- **object**
  - **AecDisplaySets** The object this property applies to.

- **ViewPort**
  - AcadObject; input-only
  - The AcadViewport or AcadPViewport that the display configuration applies to.

- **RetVal**
  - **AecDisplaySet**
  - The active display set for the specified viewport.
**AdjustSizing Property**

Specifies whether you can override the default offset values of the anchored object, thus changing its size.

See Also | [Example](#)

**Signature**

```
object.AdjustSizing
```

object

*AecAnchorEntToGridAssembly* The object this property applies to.

AdjustSizing

Boolean; read-write

TRUE: Offsets can be modified.

FALSE: Offsets cannot be modified.
AllowByMaterial Property

Determines if the ByMaterial property should be applied.

See Also | Example

Signature

object.AllowByMaterial

object

AecDisplayComponentEntity The object this property applies to.

AllowByMaterial

Boolean; read-write

Remarks

This is an advanced setting that should rarely need to be changed.
AllowVariation Property

Specifies if the anchored object is allowed to vary.

See Also | Example

**Signature**

object.AllowVariation

object

*AecAnchorEntToGridAssembly* The object this property applies to.

AllowVariation

Boolean; read-write

TRUE: Allow variation.
FALSE: Don't allow variation.
**AlternateName Property**

The AlternateName of the Style.

See Also | Example

**Signature**

```
object.AlternateName
```

```
object

AecMaskBlockStyle, AecLayerKeyStyle, AecDictRecord, AecMVBloc
AecPolygonStyle, AecProfileStyle
```

The object this property applies to.

```
AlternateName

String; read-write
```

The AlternateName of the Style.
AlwaysImportLayerStandard Property

Specifies whether to always import a newer layer standard.

See Also | Example

**Signature**

object.AlwaysImportLayerStandard

object

[AecBaseDatabasePreferences](#) The object this property applies to.

AlwaysImportLayerStandard

Boolean; read-write

Specifies whether to always import a newer layer standard.

**Angle Property**

Specifies the angle of the layout grid, or the angle of the hatch pattern for a display component.

**Signature**

object.Angle

object

AecDisplayComponentHatch, AecLayoutGrid2D
The object this property applies to.

Angle

ACAD_ANGLE; read-write
The angle of the layout grid, or the angle of the hatch pattern for a display component.
Angle1 Property

Deviation of cut plane at starting edge where zero is perpendicular to the clip line.

See Also | Example

Signature

object.Angle1

  object

    AecClipVol The object this property applies to.

  Angle1

    Double (radians); read-write

Remarks

The angle is specified in radians.
**Angle2 Property**

Deviation of cut plane at ending edge where zero is perpendicular to the clip Line.

**Signature**

object.Angle2

object

[AecClipVol](#) The object this property applies to.

Angle2

Double (radians); read-write

**Remarks**

The angle is specified in radians.
**AngleFromNode Property**

Specifies the angle from the anchored entity to the node.

See Also | Example

**Signature**

object.AngleFromNode

object

**AecAnchorLeadEntToNode** The object this property applies to.

AngleFromNode

Double; read-write

The angle, in radians, from the anchored entity to the node.
AngularAzimuth Property

The format used to display angles.

See Also | Example

Signature

object.AngularAzimuth

object

AecBaseDatabasePreferences The object this property applies to.

AngularAzimuth

Long; read-write

0: Use Bearings

1: Use North Azimuths

2: Use South Azimuths
AngularDisplayFormat Property

Specifies the unit used when angular measurements are displayed.

See Also | Example

Signature

object.AngularDisplayFormat

object

AecBaseDatabasePreferences The object this property applies to.

AngularDisplayFormat

Long; read-write

0: Use Degrees

1: Use Degrees/Minutes/Seconds

2: Use Grads

3: Use Radians

4: Use Surveyor

Remarks
When you type degrees at AutoCAD 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.

See Also | Example

Signature

object.AngularPrecision

object

AecBaseDatabasePreferences The object 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 commands always calculate all numbers up to the highest internal precision.
Application Property

Gets the Application 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 application's frame controls and path settings, and provides the means to navigate down the object hierarchy.
**ApplyResize Property**

Specifies if the anchored object is resized when the layout cell is resized.

**Signature**

```plaintext
object.ApplyResize
```

- `object` is the object this property applies to.
- `ApplyResize` is a Boolean; read-write. TRUE: Apply resize. FALSE: Don't apply resize.

**Remarks**

If you want to set the `ResizeOffset` property, you must first set `ApplyResize` to True in order for the specified offset to take affect.

See Also | Example
Area Property

Gets the area of an object.

See Also | Example

Signature

object.Area

object

AecProfile, AecSlice

The object this property applies to.

Area

Double; read-only

The area enclosed by the object.
AreaDisplayUnit Property

The unit used when area measurements are displayed.

See Also | Example

Signature

object.AreaDisplayUnit

object

AecBaseDatabasePreferences The object this property applies to.

AreaDisplayUnit

AecBuiltInUnit enum; read-write
**AreaPrecision Property**

The precision to which area measurements are displayed.

See Also | Example

**Signature**

object.AreaPrecision

object

[AecBaseDatabasePreferences](#) The object 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.
AreaSuffix Property

The suffix displayed after an area unit in the drawing.

See Also | Example

Signature

object.AreaSuffix

object

AecBaseDatabasePreferences The object this property applies to.

AreaSuffix

String; read-write
The suffix displayed following the area unit.

Remarks

The default value is "Sq.Ft." or "m2", depending on the value of the MeasurementUnit property. The value must be no longer than 255 characters in length.
**BasePoint Property**

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

See Also | [Example](#)

**Signature**

`object.BasePoint`

*object*  

*AecBaseDatabasePreferences* The object 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

AecBaseDatabasePreferences The object 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.
**BayAngle Property**

Specifies the bay angle of the layout grid.

**Signature**

object.BayAngle

object

`AecLayoutGrid2D` The object this property applies to.

BayAngle

`ACAD_ANGLE`; read-write

The bay angle of the layout grid.
Block Property

Gets or sets the AcadBlock for the view.

See Also | Example

Signature

object.Block

object

AecViewBlockDef The object this property applies to.

Block

AcadBlock; read-write
## BlockBasedLayerOffBehavior Property

Immediate and independent layer on/off control of display components.

**Signature**

```plaintext
object.BlockBasedLayerOffBehavior

object

AecBaseDatabasePreferences The object this property applies to.

BlockBasedLayerOffBehavior

Boolean; read-write
Immediate and independent layer on/off control of display components.
```
Blocks Property

Gets the Blocks collection for the drawing.

See Also | Example

Signature

object.Blocks

object

AecBaseDatabase, AecDatabase
The object or objects this property applies to.

Blocks

Blocks collection; read-only
The Blocks collection for the drawing.
**BottomOffset Property**

Specifies the bottom offset of the anchored object.

See Also | Example

**Signature**

object.BottomOffset

object

AecAnchorEntToGridAssembly The anchor object this property applies to.

BottomOffset

Double; read-write

The bottom offset of the anchored object. (See Remarks.)

**Remarks**

To change the BottomOffset value, you must first set AdjustSizing to True. Modifying BottomOffset when AdjustSizing is False does not result in an error or warning, but it has no effect.
**Boundary Property**

Specifies the boundary object attached to the grid.

**Signature**

object.Boundary

object

[AecLayoutGrid2D](#) The object this property applies to.

Boundary

AcadEntity; read-write

The boundary object attached to the grid.
**Bounds Property**

Gets the upper left and lower right corners of the profile.

See Also | [Example](#)

**Signature**

object.Bounds

object

AecProfile The object this property applies to.

Bounds

Variant (array of 4 doubles); read-only
Top left / Bottom right corner of profile.
ByMaterial Property

Specifies if the display properties are determined by the material assigned to the display component.

See Also | Example

Signature

object.ByMaterial

object

AecDisplayComponentEntity The object this property applies to.

ByMaterial

Boolean; read-write
**Cell Property**

Specifies the grid cell to anchor the entity to.

See Also | [Example](#)

**Signature**

```plaintext
object.Cell

object

AecAnchorEntToGridAssembly The object this property applies to.

Cell

Long; read-write

The index of the grid cell the object is anchored to.
**CenterOnNode Property**

Specifies if the anchored object is centered on the layout node.

See Also | Example

**Signature**

object.CenterOnNode

object

[AecAnchorEntToLayoutVolume, AecAnchorLeadEntToNode, AecAnchorEntToLayoutNode, AecAnchorEntToLayoutCell]

The object this property applies to.

CenterOnNode

Boolean; read-write

TRUE: Center on node.
FALSE: Don't center on node.
Centroid Property

Gets the center of area or mass for a profile shape.

See Also | Example

Signature

object.Centroid

object

AecProfile The object this property applies to.

Centroid

Variant (two-element array of doubles); read-only Centroid point of the profile.
**ClassName Property**

The class name of the object.

**Signature**

object.ClassName

```
object
  AecDisplayRepresentation The object this property applies to.

ClassName
  String; read-write
```
ClipVol Property

Specifies the clip volume of the 2d section.

Signature

object.ClipVol

Object

Aec2dSection The object this property applies to.

ClipVol

AecClipVol; read-write
ContourLinesPerSurface Property

Specifies the number of contour lines (isolines) per surface on objects.

See Also | Example

Signature

object.ContourLinesPerSurface

object

AecBaseDatabasePreferences The object or objects this property applies to.

ContourLinesPerSurface

Integer; read-write
The valid range is 0 to 2047.

System variables

The value of this property is stored in the ISOLINES system variable.

Remarks

The initial value for this property is 4.

A torus with ContourLinesPerSurface set to 4

A torus with ContourLinesPerSurface set to 20
CoordinatePrecision Property

Specifies the precision to which coordinate measurements are displayed.

See Also | Example

Signature

object.CoordinatePrecision

object

AecBaseDatabasePreferences The object 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.
**Coordinates Property**

Returns the vertices of a ring or profile.

**Signature**

object.Coordinates

```
object

AecRing, AecProfile
The object this property applies to.
```

**Coordinates**

```
Variant (n-element array of doubles);
read-only
List of 2D points.
The vertices of this ring or profile.
```
**Count Property**

Gets the number of items in a collection, or the number of nodes in a layout curve.

See Also | Example

**Signature**

```
object.Count
```

`object`

*All Collections, AecLayoutCurve, AecDisplayRepresentations, AecDisplayComponents, AecMVBlockDisplayRepDefs, AecViewBlockDefs, AecsConnectionNodes*

The object this property applies to.

```
Count
```

Long; read-only

The number of items.
CreateDimscaleOverride Property

Automatically create a dimscale override.

Signature

object.CreateDimscaleOverride

object

AecBaseDatabasePreferences The object this property applies to.

CreateDimscaleOverride

Boolean; read-write

Automatically create a dimscale override.
CutPlaneDisplayAbove Property

Gets or sets the upper range above the cut plane to display objects.

See Also | Example

Signature

object.CutPlaneDisplayAbove

object

AecDisplayConfiguration The object this property applies to.

CutPlaneDisplayAbove

Double; read-write
**CutPlaneDisplayBelow Property**

Gets or sets the lower range below the cut plane to display objects.

See Also | Example

**Signature**

object.CutPlaneDisplayBelow

object

AecDisplayConfiguration The object this property applies to.

CutPlaneDisplayBelow

Double; read-write
CutPlaneHeight Property

Gets or sets the height of the cut plane.

See Also | Example

Signature

object.CutPlaneHeight

object AecDisplayConfiguration The object this property applies to.

CutPlaneHeight Double; read-write
CutProfile Property

Returns the cut profile for the object.

See Also | Example

Signature

object.CutProfile

object

AecMaskBlockRef The object this property applies to.

CutProfile

AecProfile; read-only
**Database Property**

Gets the database in which the object belongs.

**Signature**

`object.Database`

- `object`

  AecBaseDatabasePreferences, AecMassGroups
  The object or objects this property applies to.

**Database**

AcadDatabase object; read-only
The Database object that contains the object.
DatabaseScale Property

Specifies the database scale at which the horizontal axis of the drawing is displayed.

See Also | Example

Signature

object.DatabaseScale

object

AecBaseDatabasePreferences The object 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.
**Depth Property**

Gets or sets the size of the grid or mass element in its relative Y direction.

See Also | Example

**Signature**

object.Depth

object

AecLayoutGrid2D, AecLayoutGrid3D, AecMassElement

The object this property applies to.

Depth

Double; read-write

The size of the object in its relative Y direction; see Remarks.

**Remarks**

For radial 2D layout grids, Depth is an angle, in radians.
**DefaultDisplayProperties Property**

The collection of default display properties for the display representation.

See Also | Example

**Signature**

object.DefaultDisplayProperties

object

[AecDisplayRepresentation](#) The object this property applies to.

DefaultDisplayProperties

[AecDisplayProperties](#); read-write
**Description Property**

Gets or sets text describing the AEC Object.

**Signature**

```csharp
object.Description
```

object

*AecAnchorExtendedTagToEnt AecObject* The object this property applies to.

**Description**

String; read-write

Description of the AEC Object.
**Deviation Property**

Specifies the deviation of the mass element.

See Also | Example

**Signature**

object.Deviation

object

[AecMassElement](#) The object this property applies to.

Deviation

Double; read-write
**Dictionaries Property**

Gets the Dictionaries collection for the document.

See Also | Example

**Signature**

object.Dictionaries

object

[AecBaseDatabase, AecDatabase]
The object or objects this property applies to.

Dictionaries

[Dictionaries] collection; read-only
The Dictionaries collection for the document.
### DimStyles Property

Gets the DimStyles collection for the document.

**Signature**

```plaintext
object.DimStyles
```

object  

- **AecBaseDatabase**, **AecDatabase**
  The object or objects this property applies to.

- **DimStyles**
  
  **AcadDimStyles** collection; read-only
  The DimStyles collection for the document.

**See Also** | [Example](#)
DisplayComponents Property

Gets the collection of display components for the display representation.

See Also | Example

Signature

object.DisplayComponents

object

AecDisplayProperties The object this property applies to.

DisplayComponents

AecDisplayComponents; read-write

Remarks

The display components collection contains AecDisplayComponentEntity and AecDisplayComponentHatch objects.
**DisplayConfigurations Property**

Returns the display configurations collection.

See Also | Example

**Signature**

object.DisplayConfigurations

object

[AecBaseDatabase](#), [AecBaseDocument](#)

The object this property applies to.

DisplayConfigurations

[AecDisplayConfigurations](#); read-write
**DisplayHiddenLinework Property**

Specifies whether hidden linework is displayed.

**Signature**

object.DisplayHiddenLinework

object

[AecDisplayPropertiesMaterial] The object this property applies to.

DisplayHiddenLinework

Boolean; read-write
**DisplayName Property**

The name displayed for the object.

See Also | Example

**Signature**

object.DisplayName

object

AecDisplayRepresentation, AecMVBlocDisplayRepDef

The object this property applies to.

DisplayName

String; read-write
DisplayRepDefs Property

Returns the display representation definitions for the multi-view block style.

See Also | Example

Signature

object.DisplayRepDefs

object

AecMVBlockStyle The object this property applies to.

MVBlockDisplayRepDefs

AecMVBlockDisplayRepDefs; read-write
**DisplayRepresentations Property**

Gets the display representations collection.

**Signature**

```plaintext
object.DisplayRepresentations

object

AecDisplaySet, AecBaseDatabase, AecBaseDocument
The object this property applies to.

DisplayRepresentations

AecDisplayRepresentations; read-write
```

**Remarks**

You can append to or remove from the DisplayRepresentations collection when accessed from the DisplaySet.
DisplaySet Property

Gets or sets the display set for a given view direction.

See Also | Example

Signature

RetVal = object.DisplaySet(ViewDir)

object

AecDisplayConfiguration The object this property applies to.

ViewDir

AecViewDirection; input-only
The view direction.

RetVal

AecDisplaySet
The display set that applies used for the view direction.
DisplaySets Property

Returns the collection of display sets.

See Also | Example

Signature

object.DisplaySets

object

AecBaseDatabase, AecBaseDocument
The object this property applies to.

DisplaySets

AecDisplaySets; read-write
DisplaySilhouette Property

Controls if silhouette curves of solid objects are displayed in Wireframe mode.

See Also | Example

Signature

object.DisplaySilhouette

object

AecBaseDatabasePreferences The object or objects this property applies to.

DisplaySilhouette

Boolean; read-write

TRUE: Display only silhouette lines for solid objects.

FALSE: Display isolines for solid objects.

System variables

The value of this property is stored in the DISPSILH system variable.

Remarks

The initial value for this property is FALSE.
**Document Property**

Gets the document (drawing) in which the object belongs.

See Also | Example

**Signature**

object.Document

object

- AecAnchor
- AecAnchorEntToCurve
- AecAnchorEntToGridAssembly
- AecAnchorEntToLayoutCell
- AecAnchorEntToLayoutNode
- AecAnchorEntToLayoutVolume
- AecAnchorExtendedTagToEnt
- AecAnchorLeadEntToNode
- AecAnchorToRef
- AecBlockRef
- AecCamera
- AecCellLayoutTool
- AecClipVol
- AecClipVolRes
- AecDictionary
- AecDictRecord
- AecEntity
- AecEntRef
- AecGeo
- AecGridAssembly
- AecLayerKeyStyle
- AecLayerKeyStyles
- AecLayoutCurve
- AecLayoutGrid2D
- AecLayoutGrid3D
- AecLayoutTool
- AecMaskBlockRef
- AecMaskBlockStyle
- AecMaskBlockStyles
- AecMassElement
- AecMassGroup
- AecMVBlockRef
- AecMVBlockStyle
- AecMVBlockStyles
- AecObject
- AecPolygon
- AecPolygonStyle
- AecPolygonStyles
- AecProfileStyle
- AecProfileStyles
- AecSlice
- AecVolumeLayoutTool

The object or objects this property applies to.

**Document**

Document object; read-only

The document (drawing) that contains the object.
DoubleHatch Property

Specifies if double hatching is used.

Signature

object.DoubleHatch

object

AecDisplayComponentHatch The object this property applies to.

DoubleHatch

Boolean; read-write

Remarks

The DoubleHatch property applies when the HatchType is set to aecHatchTypeUserDefined.
**EdgeWidth Property**

Specifies the edge width of the polygon style.

See Also | [Example](#)

**Signature**

object.EdgeWidth

object

**AecPolygonStyle** The object this property applies to.

EdgeWidth

Double; read-write

The edge width of the polygon style.
**EdgeWidthJustify Property**

Specifies whether edges are displayed inside, outside, or centered on the AEC Polygon vertices.

See Also | [Example](#)

**Signature**

```
object.EdgeWidthJustify
```

object

[AecPolygonStyle](#) The object this property applies to.

**EdgeWidthJustify**

```
AecPolygonStyleEdgeWidthJustification enum;
read-write
```

The edge justification of the polygon style.
ElevationModelSpace Property

Specifies the elevation setting in the model space.

See Also | Example

Signature

object.ElevationModelSpace

object

AecBaseDatabase, AecDatabase
The object or objects this property applies to.

ElevationModelSpace
Double; read-write
The elevation setting for model space.

Remarks

The current elevation is the Z value that is used whenever a 3D point is expected but only the X and Y values are supplied. The current elevation is maintained separately in model space and paper space.
ElevationPaperSpace Property

Specifies the elevation setting in the paper space.

See Also | Example

Signature

object.ElevationPaperSpace

object

AecBaseDatabase, AecDatabase
The object or objects this property applies to.

ElevationPaperSpace

Double; read-write
The elevation for paper space.

Remarks

The current elevation is the Z value that is used whenever a 3D point is expected but only the X and Y values are supplied. The current elevation is maintained separately in model space and paper space.
**ElevationPrecision Property**

Specifies the precision to which elevation measurements are displayed in the drawing.

See Also | Example

**Signature**

object.ElevationPrecision

object

  AecBaseDatabasePreferences The object 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.
EndOffset Property

Specifies the offset distance from the end point of the curve, which may be used to reduce the effective layout curve length.

See Also | Example

**Signature**

```csharp
object.EndOffset

object

AecLayoutCurve The object this property applies to.

EndOffset

Double; read-write
Offset distance from end point of the curve.
**ExcludeFrom2dSectionShrinkwrap Property**

Specifies whether the material is excluded from the shrinkwrap of a 2D section.

**Signature**

```plaintext
object.ExcludeFrom2dSectionShrinkwrap

object

AecDisplayPropertiesMaterial The object this property applies to.

ExcludeFrom2dSectionShrinkwrap

Boolean; read-write
```
FacetDeviation Property

Specifies the allowed deviation when facetting curved lines and surfaces.

See Also | Example

Signature

object.FacetDeviation

object

AecBaseDatabasePreferences The object this property applies to.

FacetDeviation

Double; read-write
The facet deviation of AEC 3D Objects.

System variables

The value of this property is stored in the FACETDEV system variable.

Remarks

Facet deviation is similar to FacetRes for ACIS bodies.
FacetMaximum Property

Specifies the maximum number of facets used for a circle.

See Also | Example

Signature

object.FacetMaximum

object

AecBaseDatabasePreferences The object this property applies to.

FacetMaximum

Integer; read-write
The maximum number of facets used for a circle.
**FixedViewDirection Property**

Gets or sets a fixed view direction for a display configuration.

See Also | Example

**Signature**

```
object.FixedViewDirection
```

```csharp
object

AecDisplayConfiguration The object this property applies to.

FixedViewDirection

AecFixedViewDirection; read-write
FlipX Property

Determines if the anchored object is flipped about the X axis of the curve it is anchored to.

See Also | Example

Signature

object.FlipX

object

AecAnchorEntToCurve,
AecAnchorEntToGridAssembly,
AecAnchorEntToLayoutCell,
AecAnchorEntToLayoutNode,
AecAnchorEntToLayoutVolume,
AecAnchorLeadEntToNode

The object this property applies to.

FlipX

Boolean; read-write
TRUE: Anchored object is flipped around the X axis.
FALSE: Anchored object is not flipped around the X axis.
**FlipY Property**

Determines if the anchored object is flipped about the Y axis of the curve it is anchored to.

**Signature**

object.FlipY

object

AecAnchorEntToCurve, 
AecAnchorEntToGridAssembly, 
AecAnchorEntToLayoutCell, 
AecAnchorEntToLayoutNode, 
AecAnchorEntToLayoutVolume, 
AecAnchorLeadEntToNode

The object this property applies to.

**FlipY**

Boolean; read-write
TRUE: Anchored object is flipped around the Y axis.
FALSE: Anchored object is not flipped around the Y axis.
**FlipZ Property**

Determines if the anchored object is flipped about the Z axis of the curve it is anchored to.

See Also | Example

**Signature**

object.FlipZ

*object*

-AecAnchorEntToCurve,
-AecAnchorEntToGridAssembly,
-AecAnchorEntToLayoutCell,
-AecAnchorEntToLayoutNode,
-AecAnchorEntToLayoutVolume,
-AecAnchorLeadEntToNode

The object this property applies to.

FlipZ

Boolean; read-write
TRUE: Anchored object is flipped around the Z axis.
FALSE: Anchored object is not flipped around the Z axis.
Groups Property

Gets the Groups collection for the document.

See Also | Example

Signature

object.Groups

object

AecBaseDatabase, AecDatabase
The object or objects this property applies to.

Groups

AcadGroups collection; read-only
The Groups collection for the document.
Handle Property

Gets the handle of an object.

See Also | Example

Signature

object.Handle

object

AecAnchor, AecAnchorEntToCurve, AecAnchorEntToGridAssembly, AecAnchorEntToLayoutCell, AecAnchorEntToLayoutNode, AecAnchorEntToLayoutVolume, AecAnchorExtendedTagToEnt, AecAnchorLeadEntToNode, AecAnchorToRef, AecBlockRef, AecCamera, AecCellLayoutTool, AecClipVol, AecClipVolRes, AecDictionary, AecDictRecord, AecEntity, AecEntRef, AecGeo, AecGridAssembly, AecLayerKeyStyle, AecLayerKeyStyles, AecLayoutCurve, AecLayoutGrid2D, AecLayoutGrid3D, AecLayoutTool, AecMaskBlockRef, AecMaskBlockStyle, AecMaskBlockStyles, AecMassElement, AecMassGroup, AecMVBlockRef, AecMVBlockStyle, AecMVBlockStyles, AecObject, AecPolygon, AecPolygonStyle, AecPolygonStyles, AecProfileStyle, AecProfileStyles, AecSlice, AecVolumeLayoutTool

The object or objects this property applies to.

Handle

String; read-only
The handle of the entity.
Remarks

An object ID and a unique handle are the two ways of referencing an object. A handle is persistent (stays the same) in a drawing for the lifetime of the object.

In general, use a handle unless you plan to work with certain ObjectARX functions that require an object ID.
HasExtensionDictionary Property

Determines if the object has an extension dictionary associated with it.

See Also | Example

Signature

object.HasExtensionDictionary

Object

AecAnchor, AecAnchorEntToCurve, AecAnchorEntToGridAssembly, AecAnchorEntToLayoutCell, AecAnchorEntToLayoutNode, AecAnchorEntToLayoutVolume, AecAnchorExtendedTagToEnt, AecAnchorLeadEntToNode, AecAnchorToRef, AecBlockRef, AecCamera, AecCellLayoutTool, AecClipVol, AecClipVolRes, AecDictionary, AecDictRecord, AecEntity, AecEntRef, AecGeo, AecGridAssembly, AecLayerKeyStyle, AecLayerKeyStyles, AecLayoutCurve, AecLayoutGrid2D, AecLayoutGrid3D, AecLayoutTool, AecMaskBlockRef, AecMaskBlockStyle, AecMaskBlockStyles, AecMassElement, AecMassGroup, AecMVBlockRef, AecMVBlockStyle, AecMVBlockStyles, AecObject, AecPolygon, AecPolygonStyle, AecPolygonStyles, AecProfileStyle, AecProfileStyles, AecSlice, AecVolumeLayoutTool

The object or objects this property applies to.

HasExtensionDictionary

Boolean; read-only

TRUE: The object has an extension dictionary associated
with it.

FALSE: The object does not have an extension dictionary associated with it.

Remarks

You can create an extension dictionary for an object, or query an existing extension dictionary by using the `GetExtensionDictionary` method.
HatchType Property

Gets or sets the hatch type for the display component.

Signature

object.HatchType

object

[AecDisplayComponentHatch](#) The object this property applies to.

HatchType

[AecHatchType](#); read-write
**Height Property**

Specifies the height of the object.

See Also | [Example](#)

**Signature**

object.Height

object

AecClipVol, AecLayoutGrid3D, AecMassElement
The object this property applies to.

Height

Double; read-write
The relative height of the object.
Hyperlinks Property

Gets the Hyperlinks collection for an entity.

See Also | Example

**Signature**

object.Hyperlinks

object


The object or objects this property applies to.

Hyperlinks

AcadHyperlinks; read-only
The Hyperlinks collection for the entity.
Index Property

Returns the position of the item within its collection.

See Also | Example

Signature

object.Index

object

AecRing AecLayerOverrideSetting, AecLayoutCurveNode, AecLayoutGrid2DNode, AecLayoutGrid2DXNode, AecLayoutGrid2DYNode, AecLayoutGrid3DNode, AecLayoutGrid3DXNode, AecLayoutGrid3DYNode, AecLayoutGrid3DZNode, AecViewBlock, AecLayerKey, AecDisplayComponent, AecMVBlockDisplayRepDef, AecViewBlockDef, AecsConnectionNode

The object this property applies to.

Index

Long, except for AecLayerOverrideSetting, which is a String; read-only

The position of the object within its collection.

Remarks

AecLayerOverrideSetting is indexed randomly, by name.
**InsertionPoint Property**

Specifies the insertion point of the profile.

**Signature**

`object.InsertionPoint`

- `object`  
  - `AecEditInPlaceProfile` The object this property applies to.

- `InsertionPoint`  
  - `Variant (three-element array of doubles); read-write`
  - The insertion point of the profile.
**InsideRadius Property**

Specifies the inside radius of the layout grid.

**Signature**

object.InsideRadius

object

*AecLayoutGrid2D* The object this property applies to.

InsideRadius

Double; read-write
The inside radius of the layout grid.
**InterferenceBlockName Property**

Specifies the name of the interference block for the Multi-view block.

**Signature**

object.InterferenceBlockName

object

AecMVBlockStyle The object this property applies to.

InterferenceBlockName

String; read-write
Keys Property

Returns the collection of layer keys in a layer key style.

See Also | Example

Signature

object.Keys

object

AecLayerKeyStyle The object this property applies to.

Keys

AecLayerKeys; read-only
Returns the collection of AecLayerKeys.
Layer Property

Specifies the layer the object is on.

See Also | Example

Signature

object.Layer

object


The object this property applies to.

Layer

String; read-write

The name of the object's layer.
LayerFile Property

Specifies the name of the file containing default layer standard and key style definitions.

See Also | Example

Signature

object.LayerFile

object

AecBaseDatabasePreferences The object 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.
LayerKeyStyle Property

Specifies the layer key style for the object.

See Also | Example

Signature

object.LayerKeyStyle

object

AecLayerKey, AecLayerKeys, AecLayerOverrideSetting, AecLayerOverrideSettings

The object this property applies to.

LayerKeyStyle

AecLayerKeyStyle; read-write

Layer key style of the object.
Layers Property

Gets the Layers collection for the document.

See Also | Example

**Signature**

object.Layers

object

AecBaseDatabase, AecDatabase
The object or objects this property applies to.

Layers

AcadLayers collection; read-only
The Layers collection for the document.
**LayerStandard Property**

The layer standard to apply to the drawing.

See Also | [Example](#)

**Signature**

object.LayerStandard

object

[AecBaseDatabasePreferences](#) The object 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.
**LayoutCurve Property**

Provides access to layout curve geometry.

See Also | [Example](#)

**Signature**

object(LayoutCurve

object

[AecLayoutCurveNode](#), [AecLayoutCurveNodes](#)

The object this property applies to.

LayoutCurve

[AecLayoutCurve](#); read-write

A layout curve object.
**LayoutGrid2D Property**

Identifies the 2D layout grid that owns the object.

See Also | Example

**Signature**

object(LayoutGrid2D)

object

AecLayoutGrid2DNode, AecLayoutGrid2DNodes, AecLayoutGrid2DXNode, AecLayoutGrid2DXNodes, AecLayoutGrid2DYNode, AecLayoutGrid2DYNodes

The object this property applies to.

LayoutGrid2D

AecLayoutGrid2D; read-write

The layout grid that owns the layout grid node or collection of nodes.
**LayoutGrid3D Property**

Identifies the 3D layout grid that owns the object.

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

**Signature**

object/LayoutGrid3D

object


The object this property applies to.

LayoutGrid3D

[AecLayoutGrid3D](#); read-write

The 3D layout grid that owns the layout grid node or collection of nodes.
## Layouts Property

Gets the Layouts collection for the document.

### Signature

```
object.Layouts
```

- **object**: The object or objects this property applies to.
- **Layouts**: `AcadLayouts` object; read-only

The Layouts collection for the document.

See Also | [Example](#)
**LeaderExtension1 Property**

Specifies the first leader extension.

See Also | [Example](#)

**Signature**

```plaintext
object.LeaderExtension1
```

object

- [AecAnchorLeadEntToNode](#) The object this property applies to.

LeaderExtension1

- Double; read-write
  The first leader extension.
**LeaderExtension2 Property**

Specifies the second leader extension.

See Also | Example

**Signature**

object.LeaderExtension2

object

AecAnchorLeadEntToNode The object this property applies to.

LeaderExtension2

Double; read-write

The second leader extension.
LeftOffset Property

Specifies the left offset for the anchored object.

See Also | Example

Signature

object.LeftOffset

  object

    AecAnchorEntToGridAssembly The object this property applies to.

LeftOffset

  Double; read-write
  The left offset for the anchored object. (See Remarks.)

Remarks

To change the LeftOffset value, you must first set AdjustSizing to True. Modifying LeftOffset when AdjustSizing is False does not result in an error or warning, but it has no effect.
Limits Property

Specifies the drawing limits.

See Also | Example

Signature

object.Limits

object

AecBaseDatabase, AecDatabase

The object or objects this property applies to.

Limits

Variant (array of doubles); read-write
An array of four values. The first pair of values define the X and Y coordinates of the lower-left limit, the second pair of values define the X and Y coordinates of the upper-right limit.

System variables

The lower-left limit controls the LIMMIN system variable. The upper-right limit controls the LIMMAX system variable. The LIMCHECK system variable turns limits checking on and off for the current space.

Remarks

The drawing limits are two-dimensional points in the World Coordinate System (WCS) that represent a lower-left and an upper-right limit. You cannot impose limits on the Z direction.

The drawing limits also govern the portion of the drawing covered by the visible grid and determine the minimum area that the ZoomAll method displays.
**LinearDisplayFormat Property**

Specifies the format used to display linear measurement values.

**Signature**

```csharp
object.LinearDisplayFormat
```

- **object**: The object this property applies to.
- **AecBaseDatabasePreferences**: The format used to display linear measurement values. Can be one of the following:

  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

[AecBaseDatabasePreferences](#) The object 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.
LinearUnit Property

Specifies the unit used to display linear values.

See Also | Example

**Signature**

object.LinearUnit

- object

  AecBaseDatabasePreferences The object this property applies to.

- LinearUnits

  AecBuiltInUnit enum; read-write
  The unit used to display linear values. Specify one of the following:

  - aecUnitInch: Inches
  - aecUnitFoot: Feet
  - aecUnitMillimeter: Millimeters
  - aecUnitCentimeter: Centimeters
  - aecUnitDecimeter: Decimeters
  - aecUnitMeter: Meters
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.
**Linetype Property**

Specifies the linetype of the object.

See Also | Example

**Signature**

object.Linetype

**object**


The object this property applies to.

**Linetype**

String; read-write

The name of the linetype of the object.
**Linetypes Property**

Gets the Linetypes collection for the document.

**Signature**

object.Linetypes

object

AecBaseDatabase, AecDatabase
The object or objects this property applies to.

Linetypes

AcadLineTypes collection; read-only
The Linetypes collection for the document.
LinetypeScale Property

Specifies the linetype scale of an entity.

See Also | Example

Signature

object.LinetypeScale

object

AecBlockRef, AecCamera, AecCellLayoutTool,
AecClipVol, AecClipVolRes,
AecDisplayComponentEntity, AecEntity, AecEntRef,
AecGeo, AecGridAssembly, AecLayoutCurve,
AecLayoutGrid2D, AecLayoutGrid3D,
AecLayoutTool, AecMaskBlockRef,
AecMassElement, AecMassGroup,
AecMVBlockRef, AecPolygon, AecSlice,
AecVolumeLayoutTool

The object or objects this property applies to.

LinetypeScale

Double; read-write
This value must be a positive real number. The default is 1.0.

Remarks

The linetype scale of an object specifies the relative length of dash-dot linetypes per drawing unit.

Linetype scale = 1.0

Linetype scale = 0.5
Linetype scale = 0.25
LineWeight Property

Specifies the lineweight of the object.

See Also | Example

Signature

object.LineWeight

object

AecBaseDatabasePreferences, AecBlockRef, AecCamera, AecCellLayoutTool, AecClipVol, AecClipVolRes, AecDatabasePreferences, AecDisplayComponentEntity, AecEntity, AecEntRef, AecGeo, AecGridAssembly, AecLayerKey, AecLayoutCurve, AecLayoutGrid2D, AecLayoutGrid3D, AecLayoutTool, AecMaskBlockRef, AecMassElement, AecMassGroup, AecMVBlockRef, AecPolygon, AecSlice, AecVolumeLayoutTool

The object this property applies to.

LineWeight

AclLineWeight or ACAD_LWEIGHT enum; read-write Lineweight of the object.
LineweightDisplay Property

Specifies whether lineweights are displayed in model space for the drawing.

See Also | Example

Signature

object.LineweightDisplay

AecBaseDatabasePreferences The object or objects this property applies to.

Remarks

The initial value for this property is TRUE.

Regeneration time increases with lineweights that are represented by more than one pixel. Set this property to FALSE if performance slows down when working with large lineweights.
Location Property

Specifies the insertion position of the object.

See Also | Example

Signature

object.Location

object

AecBlockRef, AecCamera, AecCellLayoutTool, AecClipVol, AecClipVolRes, AecEntRef, AecGeo, AecGridAssembly, AecLayoutCurve, AecLayoutGrid2D, AecLayoutGrid3D, AecLayoutTool, AecMaskBlockRef, AecMassElement, AecMassGroup, AecMVBlockRef, AecPolygon, AecSlice, AecVolumeLayoutTool

The object this property applies to.

Location

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

Insertion position of the object.
LowerExtension Property

Specifies the distance of the lower cut plane below the clip volume position.

See Also | Example

Signature

object.LowerExtension

object

AecClipVol The object this property applies to.

LowerExtension

Double; read-write
Distance of the lower cut plane below clip volume position.
**MassElementStyles Property**

Returns the collection of mass element styles in the specified drawing database.

**Signature**

object.MassElementStyles

object

[AecBaseDatabase](#) The object this property applies to.

MassElementStyles

AecMassElementStyles; read-write

The collection of mass element styles.
MassGroup Property

Gets or sets the parent mass group of a mass element or mass group.

See Also | Example

Signature

object.MassGroup

object

AecMassElement, AecMassGroup
The object this property applies to.

MassGroup

AecMassGroup; read-write
The parent mass group, if any.
MassGroupName Property

Specifies the mass group that the mass element or group is currently attached to.

See Also | Example

Signature

object.MassGroupName

object

AecMassElement, AecMassGroup

The object this property applies to.

MassGroupName

String; read-write

Specifies the mass group that the mass element or group is currently attached to.
MaxActiveViewports Property

Specifies the maximum number of active viewports.

See Also | Example

Signature

object.MaxActiveViewports

object

AecBaseDatabasePreferences The object or objects this property applies to.

MaxActiveViewports

Integer; read-write
The maximum number of active viewports.
2 <= MaxActiveViewports <= 48

System variables

The value of this property is stored in the MAXACTVP system variable.

Remarks

Inactive viewports are blank, so their contents are not regenerated. You can, therefore, improve performance by specifying a low value. The initial value is 48.
MeasurementUnit Property

Identifies the system of units used to display measurements in the drawing.

See Also | Example

Signature

object.MeasurementUnit

object

AecBaseDatabasePreferences The object this property applies to.

MeasurementUnit

AcMeasurementUnit enum; read-only
The system of units used to display measurements.
Can be one of the following:

acEnglish: English (Imperial) units
acMetric: Metric units
# MergeCommonMaterials Property

Specifies whether common materials are merged into one component.

See Also | Example

## Signature

object.MergeCommonMaterials

object

[**AecDisplayPropertiesMaterial**](#) The object this property applies to.

MergeCommonMaterials

Boolean; read-write
ModelSpace Property

Gets the ModelSpace collection for the document.

See Also | Example

Signature

object.ModelSpace

object

AecBaseDatabase, AecDatabase
The object or objects this property applies to.

ModelSpace

AcadModelSpace collection; read-only
The ModelSpace collection for the document.
MVBlockRef Property

Gets or sets a multi-view block reference for a view block or collection of view blocks.

See Also | Example

Signature

object.MVBlockRef

object

AecViewBlock, AecViewBlocks
The object this property applies to.

MVBlockRef

AecMVBlockRef; read-write
Reference of a multi-view block inserted into a drawing.
Name Property

Returns the name of the object.

See Also | Example

Signature

object.Name

object

AecCamera, AecDictRecord, AecLayerKey, AecLayerKeyStyle, AecLayerOverrideSetting, AecMaskBlockStyle, AecMassGroup, AecMVBLockStyle, AecPolygonStyle, AecProfileStyle, AecViewBlock, AecDisplayComponent

The object this property applies to.

Name

String; read-only for all objects except AecCamera.
The name of the object.
Node Property

Specifies the layout node the anchor attaches an object to.

See Also | Example

Signature

object.Node

object

AecAnchorEntToLayoutNode,
AecAnchorLeadEntToNode,
AecAnchorEntToLayoutVolume,
AecAnchorEntToLayoutCell

The anchor object this property applies to.

Node

Long; read-write

Node entity is anchored to. The first node on the grid is 1.
Nodes Property

Returns the collection of nodes on the layout curve.

See Also | Example

Signature

object.Nodes

object

AecLayoutCurve The object this property applies to.

Nodes

AecLayoutNodes; read-only
The collection of nodes on the curve.
**NorthRotation Property**

Specifies the angle aligned to north within a drawing.

See Also | [Example](#)

**Signature**

object.NorthRotation

object

AecBaseDatabasePreferences The object 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.
**ObjectId Property**

Gets the object ID of the object.

**Signature**

```markdown
object.ObjectId
```

Object or objects this property applies to.

**ObjectId**

Long; read-only

The object ID of an entity.
Remarks

An object ID and a unique handle are both ways of referencing an object.

In general, use a handle unless you plan to work with certain ObjectARX functions that require an object ID.
Object Name Property

Gets the AutoCAD class name of the object.

See Also | Example

**Signature**

object.ObjectName

object

AecAnchor, AecAnchorEntToCurve, AecAnchorEntToGridAssembly, AecAnchorEntToLayoutCell, AecAnchorEntToLayoutNode, AecAnchorEntToLayoutVolume, AecAnchorExtendedTagToEnt, AecAnchorLeadEntToNode, AecAnchorToRef, AecBlockRef, AecCamera, AecCellLayoutTool, AecClipVol, AecClipVolRes, AecDictionary, AecDictRecord, AecEntity, AecEntRef, AecGeo, AecGridAssembly, AecLayerKeyStyle, AecLayerKeyStyles, AecLayoutCurve, AecLayoutGrid2D, AecLayoutGrid3D, AecLayoutTool, AecMaskBlockRef, AecMaskBlockStyle, AecMaskBlockStyles, AecMassElement, AecMassGroup, AecMVBlockRef, AecMVBlockStyle, AecMVBlockStyles, AecObject, AecPolygon, AecPolygonStyle, AecPolygonStyles, AecProfileStyle, AecProfileStyles, AecSlice, AecVolumeLayoutTool

The object or objects this property applies to.

**ObjectName**

String; read-only

The AutoCAD class name of an entity.
ObjectSortByPlotting Property

Toggles sorting of drawing objects by plotting order.

See Also | Example

Signature

object.ObjectSortByPlotting

object

AecBaseDatabasePreferences The object or objects this property applies to.

ObjectSortByPlotting

Boolean; read-write

TRUE: Enable sort by plotting order.

FALSE: Disable sort by plotting order.

System variables

The value of this property is stored in the SORTENTS system variable.

Remarks

The initial value of this property is TRUE.

Initially, sorting is enabled for plotting and PostScript output only. Setting additional sorting options can result in slower regeneration and redrawing times.
ObjectSortByPSOutput Property

Toggles sorting of drawing objects by PostScript output order.

See Also | Example

Signature

object.ObjectSortByPSOutput

object

AecBaseDatabasePreferences The object or objects this property applies to.

ObjectSortByPSOutput

Boolean; read-write

TRUE: Enable sort by PostScript output order.

FALSE: Disable sort by PostScript output order.

System variables

The value of this property is stored in the SORTENTS system variable.

Remarks

The initial value of this property is TRUE.

Initially, sorting is enabled for plotting and PostScript output only. Setting additional sorting options can result in slower regeneration and redrawing times.
ObjectSortByRedraws Property

Toggles sorting of drawing objects by redraw order.

See Also | Example

Signature

object.ObjectSortByRedraws

object

AecBaseDatabasePreferences The object or objects this property applies to.

ObjectSortByRedraws

Boolean; read-write

TRUE: Enable sort by redraw order.

FALSE: Disable sort by redraw order.

System variables

The value of this property is stored in the SORTENTS system variable.

Remarks

The initial value of this property is FALSE.

Initially, sorting is enabled for plotting and PostScript output only. Setting additional sorting options can result in slower regeneration and redrawing times.
ObjectSortByRegens Property

Toggles sorting of drawing objects by regeneration order.

See Also | Example

Signature

object.ObjectSortByRegens

object

\texttt{AecBaseDatabasePreferences} The object or objects this property applies to.

ObjectSortByRegens

Boolean; read-write

TRUE: Enable sort by regeneration order.

FALSE: Disable sort by regeneration order.

System variables

The value of this property is stored in the \texttt{SORTENTS} system variable.

Remarks

The initial value of this property is FALSE.

Initially, sorting is enabled for plotting and PostScript output only. Setting additional sorting options can result in slower regeneration and redrawing times.
ObjectSortBySelection Property

Toggles sorting of drawing objects by object selection.

See Also | Example

Signature

object.ObjectSortBySelection

object

AecBaseDatabasePreferences The object or objects this property applies to.

ObjectSortBySelection

Boolean; read-write

TRUE: Enable sort by object selection.

FALSE: Disable sort by object selection.

System variables

The value of this property is stored in the SORTENTS system variable.

Remarks

The initial value of this property is FALSE.

Initially, sorting is enabled for plotting and PostScript output only. Setting additional sorting options can result in slower regeneration and redrawing times.
ObjectSortBySnap Property

Toggles sorting of drawing objects by object snap.

See Also | Example

Signature

object.ObjectSortBySnap

object

AecBaseDatabasePreferences The object or objects this property applies to.

ObjectSortBySnap

Boolean; read-write

TRUE: Enable sort by object snap.

FALSE: Disable sort by object snap.

System variables

The value of this property is stored in the SORTENTS system variable.

Remarks

The initial value of this property is FALSE.

Initially, sorting is enabled for plotting and PostScript output only. Setting additional sorting options can result in slower regeneration and redrawing times.
Offset Property

Specifies the offset from the object.

See Also | Example

Signature

object.Offset

object

AecAnchorLeadEntToNode, AecAnchorEntToLayoutCell, AecAnchorEntToLayoutNode, AecAnchorEntToLayoutVolume, AecEntRef
The object this property applies to.

Offset

Variant; read-write
Offset from the layout node or entity reference.
OLELaunch Property

Determines whether to launch the parent application when plotting OLE objects.

Signature

object.OLELaunch

object  AecBaseDatabasePreferences  The object or objects this property applies to.

OLELaunch

Boolean; read-write

TRUE: The parent application of an OLE object is launched to print it.

FALSE: The OLE object is printed from the current application.

System variables

The value of this property is stored in the OLESTARTUP system variable.

Remarks

The initial value of this property is FALSE.

Plotting from the parent application achieves a higher quality plot; however, the speed of the plot is decreased.
Operation Property

Specifies how objects are combined.

See Also | Example

Signature

object.Operation

object

AecMassElement, AecMassGroup
The object this property applies to.

Operation

AecMassOperation enum; read-write
Boolean operation: addition, subtraction, or intersection.
OverrideDisplayProperties Property

The collection of display property overrides for the display representation.

See Also | Example

Signature

object.OverrideDisplayProperties AecObject

object

AecDisplayRepresentation The object this property applies to.

OverrideDisplayProperties

AecDisplayProperties; read-write

AecObject

AecObject; input-only
An object instance or style override.

Remarks

Overrides can be applied by object or by style.
OverridesEnabled Property

Specifies whether layer overrides are applied to the style when a layer is generated from a key.

See Also | Example

Signature

object.OverridesEnabled

object

AecLayerKeyStyle The object this property applies to.

OverridesEnabled

Boolean; read-write
TRUE: Overrides are applied.
FALSE: Overrides are not applied.
OverrideSettings Property

Returns the collection of layer key override settings for a layer key style.

See Also | Example

Signature

object.OverrideSettings

object

AecLayerKeyStyle The object this property applies to.

OverrideSettings

AecLayerOverrideSettings; read-only
The collection of layer key override settings.
OwnerID Property

Gets the object ID of the owner (parent) object.

Signature

object.OwnerID

object

AecAnchor, AecAnchorEntToCurve, AecAnchorEntToGridAssembly, AecAnchorEntToLayoutCell, AecAnchorEntToLayoutNode, AecAnchorEntToLayoutVolume, AecAnchorExtendedTagToEnt, AecAnchorLeadEntToNode, AecAnchorToRef, AecBlockRef, AecCamera, AecCellLayoutTool, AecClipVol, AecClipVolRes, AecDictionary, AecDictRecord, AecEntity, AecEntRef, AecGeo, AecGridAssembly, AecLayerKeyStyle, AecLayerKeyStyles, AecLayoutCurve, AecLayoutGrid2D, AecLayoutGrid3D, AecLayoutTool, AecMaskBlockRef, AecMaskBlockStyle, AecMaskBlockStyles, AecMassElement, AecMassGroup, AecMVBlockRef, AecMVBlockStyle, AecMVBlockStyles, AecObject, AecPolygon, AecPolygonStyle, AecPolygonStyles, AecProfileStyle, AecProfileStyles, AecSlice, AecVolumeLayoutTool

The object or objects this property applies to.

OwnerID

Long; read-only
The object ID of an object's owner.

See Also | Example
PaperSpace Property

Gets the PaperSpace collection for the document.

See Also | Example

Signature

object.PaperSpace

object

AecBaseDatabase, AecDatabase
The object or objects this property applies to.

PaperSpace

AcadPaperSpace collection; read-only
The PaperSpace collection for the document.
PatternName Property

Gets or sets the name of the custom hatch pattern.

See Also | Example

Signature

object.PatternName

object

AecDisplayComponentHatch The object this property applies to.

PatternName

String; read-write

Remarks

The PatternName applies when the HatchType is set to aecHatchTypeCustomDefined.
Perimeter Property

Specifies the perimeter of the profile.

See Also | Example

Signature

object.Perimeter

object

AecProfile The object this property applies to.

Perimeter

Double; read-only
**PlaneDepth Property**

Specifies the plane depth of the slice.

**Signature**

object.PlaneDepth

object

[AecSlice](#) The object this property applies to.

PlaneDepth

Double; read-write

See Also | Example
**PlaneWidth Property**

Specifies the plane width of the slice.

**Signature**

```plaintext
object.PlaneWidth
```

`object`  
`AecSlice` The object this property applies to.

`PlaneWidth`  
Double; read-write
**PlotConfigurations Property**

Gets the PlotConfigurations collection for the document.

See Also | Example

**Signature**

object.PlotConfigurations

object

AecBaseDatabase, AecDatabase
The object or objects this property applies to.

PlotConfigurations

AcadPlotConfigurations collection; read-only
The PlotConfigurations collection for the document.
**PlotStyleName Property**

Specifies the plotstyle name for the object.

See Also | [Example](#)

**Signature**

object.PlotStyleName

object


The object this property applies to.

**PlotStyleName**

String; read-write

The name of the object's plotstyle.
Plottable Property

Specifies whether the layer generated from the key is plottable.

See Also | Example

Signature

object.Plottable

object

\texttt{AecLayerKey} The object this property applies to.

Plottable

Boolean; read-write
TRUE: Layer is plottable
FALSE: Layer is not plottable
PolygonStyles Property

Returns the collection of polygon styles in the specified drawing database.

See Also | Example

Signature

object.PolygonStyles

object

AecBaseDatabase The object this property applies to.

PolygonStyles

AecPolygonStyles; read-only
The collection of polygon styles.
Preferences Property

Returns the Preferences object for the specified drawing.

See Also | Example

Signature

object.Preferences

object

AecBaseDatabase, AecDatabase
The object this property applies to.

Preferences

AcadDatabasePreferences; read-only
The Preferences object for the drawing.
Profile Property

Returns or sets the object's profile.

See Also | Example

Signature

object.Profile

object

\[ \text{AecEditInPlaceProfile, AecPolygon, AecProfileStyle, AecRing, AecRings, AecMassElement, AecSlice} \]

The object this property applies to.

Profile

\[ \text{AecProfile}; \text{read-write} \]

The object's profile.
**ProfileStyle Property**

Specifies the profile style of the mass element.

**Signature**

```
object.ProfileStyle
```

- **object**: `AecMassElement` The object this property applies to.

- **ProfileStyle**: `AecProfileStyle`; read-write
  The profile style of the mass element.

See Also | [Example](#)
ProfileStyleName Property

Specifies the name of the object's profile style.

See Also | Example

Signature

object.ProfileStyleName

object

AecEditInPlaceProfile, AecMassElement
The objects this property applies to.

ProfileStyleName

String; read-write
The name of the profile style.
ProjectName Property

Specifies the name of the project that the drawing belongs to.

See Also | Example

Signature

object.ProjectName

  object  AecBaseDatabasePreferences  The object this property applies to.

  ProjectName

      String; read-write
  The project name.

Remarks

The project name can be no more than 255 characters in length.
Radius Property

Specifies the radius of curved mass elements.

See Also | Example

Signature

object.Radius

  object

    AecMassElement The object this property applies to.

  Radius

    Double; read-write
    The radius of the mass element.
**Reference Property**

Specifies the object being anchored to.

See Also | [Example](#)

**Signature**

object.Reference

object

- [AecAnchorEntToLayoutCell](#), [AecAnchorEntToCurve](#), [AecAnchorLeadEntToNode](#), [AecAnchorEntToGridAssembly](#), [AecAnchorEntToLayoutNode](#), [AecAnchorEntToLayoutVolume](#), [AecAnchorExtendedTagToEnt](#), [AecAnchorToRef](#)

The object this property applies to.

**Reference**

AcadEntity; read-write, except for [AecAnchorEntToGridAssembly](#), which is read-only.

The object being anchored to.
RegisteredApplications Property

Gets the RegisteredApplications collection for the document.

See Also | Example

Signature

object.RegisteredApplications

object

AecBaseDatabase, AecDatabase
The object or objects this property applies to.

RegisteredApplications

AcadRegisteredApplications collection; read-only
The RegisteredApplications collection for the document.
Removeable Property

Specifies whether you can remove the layer key from the layer key style.

See Also | Example

Signature

object.Removeable

object

AecLayerKey The object this property applies to.

Removeable

Boolean; read-only
TRUE: Layer key is removable.
FALSE: Layer key is not removable.
RenderSmoothness Property

Specifies the smoothness of shaded, rendered, and hidden line-removed objects.

See Also | [Example](#)

**Signature**

object.RenderSmoothness

object

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

RenderSmoothness

Double; read-write
The valid range is 0.01 to 10.0.

**System variables**

The value of this property is stored in the [FACETRES](#) system variable.

**Remarks**

The initial value for this property is 0.5. To improve performance, set this value to 1 or less when drawing.
ResizeOffset Property

Specifies the offset of the anchored object to the cell boundary when the cell is resized.

See Also | Example

Signature

object.ResizeOffset

object

AecAnchorEntToLayoutCell, AecAnchorEntToLayoutVolume

The object this property applies to.

ResizeOffset

Double; read-write

The offset from the cell to the anchored object.

Remarks

To change the ResizeOffset value, you must first set ApplyResize to True. Modifying ResizeOffset when ApplyResize is False does not result in an error or warning, but it has no effect.

To specify a distance between the anchored object and the cell, specify a negative value for ResizeOffset. To extend the anchored object beyond the cell edges, specify a positive value.
**RightOffset Property**

Specifies the right offset for the anchored object.

See Also | [Example](#)

**Signature**

```none
object.RightOffset
```

**object**

- `AecAnchorEntToGridAssembly` The object this property applies to.

**RightOffset**

- `Double; read-write`
  - The right offset for the anchored object.

**Remarks**

To change the RightOffset value, you must first set `AdjustSizing` to True. Modifying RightOffset when AdjustSizing is False does not result in an error or warning, but it has no effect.
**Rings Property**

Returns the collection of rings that make up the profile.

**Signature**

```
object.Rings
```

object

*AecProfile* The object this property applies to.

**Rings**

*AecRings*; read-only

Collection of rings that make up the profile.
Rise Property

Specifies the rise of gabled mass elements.

See Also | Example

Signature

object.Rise

    object

        AecMassElement The object this property applies to.

    Rise

        Double; read-write
        The rise of the gabled mass element.
Rotation Property

Specifies the rotation reference angle for the object.

See Also | Example

Signature

object.Rotation

The object this property applies to.

Rotation

ACAD_ANGLE; read-write
Rotation reference angle for the object.
**Scale Property**

Specifies the scale factor applied to the hatch pattern.

**Signature**

object.Scale

object

[AecDisplayComponentHatch](#) The object this property applies to.

**Scale**

Double; read-write

**Remarks**

The Scale property determines how the hatch pattern is repeated.
**ScaleOnInsert Property**

Specifies whether objects are automatically scaled when inserted into the drawing.

See Also | Example

**Signature**

object.ScaleOnInsert

object

[**AecBaseDatabasePreferences**](#) The object this property applies to.

ScaleOnInsert

Boolean; read-write

TRUE: New objects are scaled automatically.
FALSE: New objects are not scaled automatically.
**ScaleX Property**

Specifies the X direction scale factor of the inserted block or entity.

See Also | **Example**

**Signature**

object.ScaleX

    object

        AecBlockRef, AecEntRef, AecMaskBlockRef, AecMVBRef
        The object this property applies to.

**ScaleX**

    Double; read-write
    Scale of the block in the X direction.
**ScaleY Property**

Specifies the Y direction scale factor of the inserted block or entity.

See Also | [Example](#)

**Signature**

object.ScaleY

object

[AecBlockRef, AecEntRef, AecMaskBlockRef, AecMVBlockRef]

The object this property applies to.

ScaleY

Double; read-write

Scale of the block in the Y direction.
ScaleZ Property

Specifies the Z direction scale factor of the inserted block or entity.

See Also | Example

Signature

object.ScaleZ

object

AecBlockRef, AecEntRef, AecMaskBlockRef, AecMVBlockRef
The object this property applies to.

ScaleZ

Double; read-write
Scale of the block in the Z direction.
SectionedBodyRenderingMaterialName Property

Gets or sets the name of the sectioned body rendering material.

See Also | Example

**Signature**

object.SectionedBodyRenderingMaterialName

object

[AecDisplayPropertiesMaterial](#) The object this property applies to.

SectionedBodyRenderingMaterialName

String; read-write
SectionRenderingMaterialName Property

Gets or sets the name of the section rendering material.

See Also | Example

Signature

object.SectionRenderingMaterialName

object

AecDisplayPropertiesMaterial The object this property applies to.

SectionRenderingMaterialName

String; read-write
**SegmentPerPolyline Property**

Specifies the number of line segments to be generated for each polyline curve.

See Also | Example

**Signature**

object.SegmentPerPolyline

    object
        AecBaseDatabasePreferences The object or objects this property applies to.

    SegmentPerPolyline
        Integer; read-write
        The number of line segments to be generated for each polyline curve.
        0 < SegmentPerPolyline <= 42950

**System variables**

The value of this property is stored in the SPLINESEGS system variable.

**Remarks**

The initial value is 8. The higher the number you specify, the greater the performance impact.
SelfIntersects Property

Determines if the profile is self-intersecting.

See Also | Example

Signature

object(SelfIntersects

object

AecProfile The object this property applies to.

SelfIntersects

Boolean; read-only
TRUE: Profile intersects itself.
FALSE: Profile does not intersect itself.
## Shape Property

Specifies the shape of the layout grid.

### Signature

```plaintext
object.Shape
```

- object: `AecLayoutGrid2D` The object this property applies to.

- Shape: `AecLayoutShape` enum; read-write

The shape of the layout grid.
Side1 Property

Specifies the length of the cut plane along the starting edge.

See Also | Example

Signature

object.Side1

object

AecClipVol | The object this property applies to.

Side1

Double; read-write
The length of the cut plane along the starting edge.
**Side2 Property**

Specifies the length of the cut plane along the ending edge.

See Also | [Example](#)

**Signature**

object.Side2

object

[AecClipVol](#) The object this property applies to.

Side2

Double; read-write
Length of cut plane along the ending edge.
SolidFill Property

Specifies if multilines, traces, solids, all hatches (including solid-fill) and wide polylines are filled in.

See Also | Example

Signature

object.SolidFill

object

AecBaseDatabasePreferences The object or objects this property applies to.

SolidFill

Boolean; read-write

TRUE: Displays solid fills.

FALSE: Does not display solid fills.

System variables

The value of this property is stored in the FILLMODE system variable.

Remarks

The initial value of this property is TRUE.
**Spacing Property**

Specifies the spacing between nodes or for an individual node on a layout tool, or the spacing between repeated hatch patterns for a display component.

See Also | Example

**Signature**

object.Spacing

object


The object this property applies to.

**Spacing**

Double; read-write

The spacing between nodes or for an individual node on a layout tool, or the spacing between repeated hatch patterns for a display component.
StartOffset Property

Specifies the offset distance from the start point of the curve, which may be used to reduce the effective layout curve length.

See Also | Example

Signature

object.StartOffset

object

AecLayoutCurve The object this property applies to.

StartOffset

Double; read-write
The offset distance from the start point of the curve.
**Style Property**

Specifies the style of the object.

**Signature**

object.Style

**object**

*AecMaskBlockRef, AecMassElement, AecMVBlockRef, AecPolygon*

The object this property applies to.

**Style**

*Style; read-write*

The style of the object. The type of style depends on the type of object, as follows:

* AecMaskBlockRef  -  AecMaskBlockStyle
* AecMassElement   -  AecMassElementStyle
* AecMVBlockRef    -  AecMVBlockStyle
* AecPolygon       -  AecPolygonStyle*
StyleName Property

Specifies the name of the style that applies to the object.

See Also | Example

**Signature**

object.StyleName

object

AecMaskBlockRef, AecMVBlockRef, AecPolygon
The object this property applies to.

StyleName

String; read-write
The style name.
SubType Property

Specifies the mass element shape subtype.

See Also | Example

Signature

object.SubType

object

AecMassElement The object this property applies to.

SubType

AecMassElementSubType; read-only
SupportsAnchoring Property

Specifies whether the mass group supports element anchoring.

See Also | Example

Signature

object.SupportsAnchoring

object

AecMassGroup The object this property applies to.

SupportsAnchoring

Boolean; read-write

Remarks

This property enables enhanced mass group behavior.
SurfaceHatchPlacement Property

Specifies how the surface hatch pattern is placed on the object.

See Also | Example

Signature

object.SurfaceHatchPlacement

object

AecDisplayPropertiesMaterial The object this property applies to.

SurfaceHatchPlacement

AecSurfaceHatchPlacement; read-write
SurfaceRenderingMaterialName Property

Gets or sets the name of the surface rendering material.

See Also | Example

Signature

object.SurfaceRenderingMaterialName

object

AecDisplayPropertiesMaterial The object this property applies to.

SurfaceRenderingMaterialName

String; read-write
**SurfaceRenderMaterialMapping Property**

Specifies how the surface render material is mapped to the object.

**Signature**

```plaintext
object.SurfaceRenderMaterialMapping

object

AecDisplayPropertiesMaterial The object this property applies to.

SurfaceRenderMaterialMapping

AecSurfaceMappingType; read-write
```
TextFrameDisplay Property

Toggles the display of frames for text objects instead of displaying the text itself.

See Also | Example

Signature

object.TextFrameDisplay

object

AecBaseDatabasePreferences

The object or objects this property applies to.

TextFrameDisplay

Boolean; read-write

TRUE: Displays the frame around the extents of the text, but does not display the text itself.

FALSE: Displays the full text.

System variables

The value of this property is stored in the QTEXTMODE system variable.

Remarks

The initial value for this property is FALSE.

After you enable or disable this option, you must use the Regen method to update the display.
**TextHeight Property**

Specifies the default height of text in the drawing.

See Also | [Example](#)

**Signature**

```csharp
object.TextHeight
```

- **object**
  - [AecBaseDatabasePreferences](#) The object 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.
TextStyles Property

Gets the TextStyles collection for the document.

See Also | [Example](#)

**Signature**

object.TextStyles

object

[AcadTextStyles](#) collection; read-only

The TextStyles collection for the document.
**TopOffset Property**

Specify the top offset for the anchored object.

See Also | [Example](#)

**Signature**

object.TopOffset

    object
    AecAnchorEntToGridAssembly The object this property applies to.

    TopOffset
    Double; read-write
    The top offset for the anchored object. (See Remarks.)

**Remarks**

To change the TopOffset value, you must first set AdjustSizing to True. Modifying TopOffset when AdjustSizing is False does not result in an error or warning, but it has no effect.
Type Property

For layout curves, specifies how nodes are specified along the curve; for mass elements, specifies the shape of the mass element.

See Also | Example

Signature

object.Type

object

AecLayoutCurve, AecMassElement
The object this property applies to.

Type

AecLayoutCurve: AecLayoutType enum; read-write
Type of node spacing along the curve.

AecMassElement: AecMassElementType enum; read-write
The Mass Element shape.
UseAngleOfObject Property

Specifies the orientation of the hatch pattern.

See Also | Example

Signature

object.UseAngleOfObject

object

AecDisplayComponentHatch The object this property applies to.

UseAngleOfObject

Boolean; read-write
TRUE: Orient the hatching to the object, regardless of the object rotation.
FALSE: Orient the hatching to the World Coordinate System.
**UseFixedViewDirection Property**

Specifies if the FixedViewDirection is used for the display configuration.

See Also | Example

**Signature**

```csharp
object.UseFixedViewDirection
```

object

**AecDisplayConfiguration** The object this property applies to.

UseFixedViewDirection

Boolean; read-write
UseModelExtentsProperty

Determine if the height and lower extension are set automatically using model extents.

See Also | Example

**Signature**

object.UseModelExtents

object

AecClipVol The object this property applies to.

UseModelExtents

Boolean; read-write
TRUE: Height and Lower Extension are set automatically
FALSE: Height and Lower Extension are not set automatically
UseOffset Property

Specifies if the offset value is used.

See Also | Example

Signature

object.UseOffset

object

AecEntRef The object this property applies to.

UseOffset

Boolean; read-write
UserCoordinateSystems Property

Gets the UCSs collection for the document.

See Also | Example

Signature

object.UserCoordinateSystems

object

**AecBaseDatabase, AecDatabase**
The object or objects this property applies to.

UserCoordinateSystems

**AcadUCSs** collection; read-only
The UCSs collection for the document.
Valid Property

Tests the validity of the object.

See Also | Example

Signature

object.Valid

object

AecProfile, AecRing
The object this property applies to.

Valid

Boolean; read-only
TRUE: Object is valid.
FALSE: Object is not valid.

Remarks

A ring is valid if the following are true:

- There is more than one segment.
- The segments are continuous.
- The segments are closed.
- No segment intersects itself.

A profile is valid if each ring is closed and does not self-intersect, and the profile is not self intersecting.
**Value Property**

For layer overrides, specifies the value of the layer key style override; for layout curves, specifies either the spacing between nodes or the number of nodes on the layout curve.

See Also | Example

**Signature**

object.Value

object

AecLayerOverrideSetting, AecLayoutCurve
The object this property applies to.

**Value**

AecLayerOverrideSetting: String; read-write
The string the layer key override is set to.

AecLayoutCurve: Variant; read-write
Bay spacing or number of even divisions on a layout curve.

**Remarks**

AecLayoutCurve:

- If the curve uses manual spacing, Value is zero.
- Check the Type property to determine what Value represents.
**VerticalScale Property**

Specifies the scale at which the vertical axis of the drawing is displayed (in Land Development Desktop only).

See Also | [Example](#)

**Signature**

```plaintext
object.VerticalScale
```

- **object**
  - The object this property applies to.

**Remarks**

This is only used in Autodesk Land Development Desktop.
**ViewBlockDefs Property**

Returns the collection of view block definitions for the multi-view block display representation definition.

**Signature**

object.ViewBlockDefs

object  

*AecMVBBlockDisplayRepDef* The object this property applies to.

ViewBlockDefs  

*AecViewBlockDefs*; read-write
**ViewBlocks Property**

Returns the collection of view blocks used for display representations by the multi-view block reference.

See Also | Example

**Signature**

object.ViewBlocks

object

**AecMVBlockRef** The object this property applies to.

ViewBlocks

**AecViewBlocks**; read-only
The collection of view block references, with each multi-view block display representation mapped to an individual AcadBlock.
ViewDirection Property

Specifies the view direction for the view block definition.

Signature

object.ViewDirection

object

AecViewBlockDef The object this property applies to.

ViewDirection

AecViewDirection; read-write
**Viewports Property**

Gets the Viewports collection for the document.

See Also | Example

**Signature**

object.Viewports

object

*AecBaseDatabase, AecDatabase*

The object or objects this property applies to.

**Viewports**

*AecViewports* collection; read-only

The Viewports collection for the document.
Views Property

Gets the Views collection for the document.

See Also | Example

Signature

object.Views

object

AecBaseDatabase, AecDatabase
The object or objects this property applies to.

Views

AcadViews collection; read-only
The Views collection for the document.
Visible Property

Specifies whether the object or application is visible.

See Also | Example

Signature

object.Visible

object


The object this property applies to.

Visible

Boolean; read-only
TRUE: The object or application is visible.
FALSE: The object or application is not visible.

Remarks

Other factors can also cause an object to be invisible. For example, an object will not be displayed if its layer is off or frozen.
Specifying the application to be invisible allows you to run tasks in the background without having to see the component.
Void Property

Specifies if a ring produces a void in a profile.

See Also | Example

Signature

object.Void

object

AecRing The object this property applies to.

Void

Boolean; read-write
TRUE: Ring is void.
FALSE: Ring is additive.
Volume Property

Returns the volume of the mass element or mass group.

See Also | Example

Signature

object.Volume

object

AecMassElement, AecMassGroup
The object this property applies to.

Volume

Double; read-only
The volume of the mass element or mass group.
**VolumeDisplayUnit Property**

Specifies the units used to display volume measurements.

See Also | Example

**Signature**

object.VolumeDisplayUnit

object

AecBaseDatabasePreferences The object this property applies to.

VolumeDisplayUnit

AecBuiltInUnit enum; read-write

The units used to display volume measurements.
**VolumePrecision Property**

Specifies the number of decimal places displayed for volume measurements.

See Also | [Example](#)

**Signature**

object.VolumePrecision

    object

[AecBaseDatabasePreferences](#) The object this property applies to.

VolumePrecision

    Long; read-write

The display precision for volume measurements.

**Remarks**

The precision settings are used only for labeling and listing values, not actual computations.
**VolumeSuffix Property**

The suffix string added when volume measurements are displayed in the drawing.

**Signature**

`object.VolumeSuffix`

- `object`:
  - `AecBaseDatabasePreferences` The object this property applies to.

- `VolumeSuffix`:
  - `String; read-write`
  - The suffix displayed after the volume measurement.

**Remarks**

The default value is " Cu.Ft." or " m3", depending on the value of the `MeasurementUnit` property. The value can be up to 255 characters in length.
Width Property

Specifies the width of the object.

See Also | Example

Signature

object.Width

      object

      AecLayoutGrid2D, AecLayoutGrid3D,
      AecMassElement

The object this property applies to.

Width

      Double; read-write

      The width of the object.
WindowState Property

Specifies the state of the application or document window.

See Also | Example

Signature

object.WindowState

object

AecBaseDocument The object or objects this property applies to.

WindowState

acWindowState enum; read-write
Indicates if the application window is minimized or maximized.
WindowTitle Property

Gets the title of the document window.

See Also | Example

**Signature**

object.WindowTitle

object

AecBaseDocument The object or objects this property applies to.

WindowTitle

String; read-only
The window title of the document.
XCount Property

Specifies the number of nodes along the X direction.

See Also | Example

Signature

object.XCount

object

AecLayoutGrid2D, AecLayoutGrid3D

The object this property applies to.

XCount

Long; read-write

The number of nodes along the X direction.
XDistance Property

Specifies the distance along the curve's X axis to the anchored object.

See Also | Example

Signature

object.XDistance

object

AecAnchorEntToCurve The object this property applies to.

XDistance

Double; read-write
Distance along the curve's X axis to the anchored object.
**XOffset Property**

Specifies the offset in the X direction of the view block within the multi-view block, or the x offset for the hatch pattern of a display component.

See Also | [Example](#)

**Signature**

object.XOffset

object

AecDisplayComponentHatch, AecViewBlock

The object this property applies to.

XOffset

Double; read-write

The offset in the X direction.
XPositionFrom Property

Specifies the position on the curve from which the XDistance to an anchored object is measured.

See Also | Example

**Signature**

object.XPositionFrom

object

AecAnchorEntToCurve The object this property applies to.

XPositionFrom

AecCurvePosition enum; read-write
The position on the curve from which XDistance is measured.
**XPositionTo Property**

Specifies the position on the anchored object to measure XDistance to.

See Also | [Example]

**Signature**

object.XPositionTo

object

[AecAnchorEntToCurve] The object this property applies to.

XPositionTo

[AecEdgePosition] enum; read-write
The position that XDistance measures to.
XRefEdit Property

Determines if the current drawing can be edited in place when being referenced by another user.

See Also | Example

Signature

object.XRefEdit

object

AecBaseDatabasePreferences The object or objects this property applies to.

XRefEdit

Boolean; read-write

TRUE: Allows in-place reference editing.

FALSE: Does not allow in-place reference editing.

System variables

The value of this property is stored in the XEDIT system variable.

Remarks

The initial value of this property is TRUE.
**XRefLayerVisibility Property**

Determines the visibility of xref-dependent layers and specifies if nested xref path changes are saved.

See Also | Example

**Signature**

object.XRefLayerVisibility

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

XRefLayerVisibility  
Boolean; read-write

TRUE: Xref-dependent layer changes made in the current drawing take precedence.

FALSE: The layer table, as stored in the reference drawing (xref) takes precedence.

**System variables**

The value of this property is stored in the *VISRETAIN* system variable.

**Remarks**

The initial value of this property is TRUE.

When this property is set to TRUE, layer settings are saved with the current drawing's layer table and persist from session to session. Nested xref path changes are saved with the current drawing and persist from session to session.

When this property is set to FALSE, changes made to xref-dependent layers in the current drawing are valid in the current session only, but are
not saved with the drawing. When the current drawing is reopened, the
layer table is reloaded from the reference drawing and the current
drawing reflects those settings. The layer settings affected are: On/Off,
Freeze/Thaw, Color, and Linetype. This setting also specifies that
changes made to the paths of nested xrefs are for the current session
only and are not saved with the drawing.
XrefOverlaysUseOwnDisplayConfig Property

When used as an Xref Overlay, use this drawing's default display configuration initially.

See Also | Example

Signature

object.XrefOverlaysUseOwnDisplayConfig

object

AecBaseDatabasePreferences The object or objects this property applies to.

XrefOverlaysUseOwnDisplayConfig

Boolean; read-write

When used as an Xref Overlay, use this drawing's default display configuration initially.
### XRotation Property

Specifies the rotation of the anchored object around the X axis.

See Also | [Example](#)

#### Signature

```object.XRotation

object

AecAnchorEntToLayoutCell, AecAnchorEntToLayoutNode, AecAnchorEntToLayoutVolume, AecAnchorEntToCurve, AecAnchorLeadEntToNode

The object this property applies to.
```

XRotation

```Double; read-write
Rotation of the anchored object around the X axis.
```
XSpacing Property

Specifies the spacing between nodes along the X direction of the layout grid.

See Also | Example

Signature

object.XSpacing

object

AecLayoutGrid2D, AecLayoutGrid3D
The object this property applies to.

XSpacing

ACAD_DISTANCE; read-write
The spacing between nodes along the X direction of the layout grid.
**YAlignment Property**

Specifies the alignment of the anchored object in relation to the baseline of the assembly.

See Also | Example

**Signature**

object.YAlignment

object

*AecAnchorEntToGridAssembly* The object this property applies to.

YAlignment

*AecInfillAlignment* enum; read-write
Alignment of the infill anchored to the assembly.
(See Remarks.)

**Remarks**

To change the YAlignment value, you must first set *AllowVariation* to True. Modifying YAlignment when AllowVariation is False does not result in an error or warning, but it has no effect.
# YCount Property

Specifies the number of nodes along the Y direction.

**Signature**

```
object.YCount
```

- `object`: AecLayoutGrid2D, AecLayoutGrid3D
  The object this property applies to.

- `YCount`: Long; read-write
  The number of nodes along the Y direction.
YDistance Property

Specifies the distance along the curve's Y axis to the anchored object.

See Also | Example

Signature

object.YDistance

object

AecAnchorEntToCurve The object this property applies to.

YDistance

Double; read-write
Distance along the curve's Y axis to the anchored object.
YOffset Property

Specifies the Y offset of the anchored object, or the offset in the Y direction of the view block within the multi-view block, or the Y offset for the hatch pattern of a display component.

See Also | Example

Signature

object.YOffset

object

AecAnchorEntToGridAssembly, AecDisplayComponentHatch, AecViewBlock

The object this property applies to.

YOffset

Double; read-write

The offset in the Y direction.
YPositionFrom Property

Specifies the position on the curve from which the YDistance to an anchored object is measured.

See Also | Example

Signature

object.YPositionFrom

object

AecAnchorEntToCurve The object this property applies to.

YPositionFrom

AecCurveWidthPosition enum; read-write
The position on the curve from which YDistance is measured.
**YPositionTo Property**

Specifies the position on the anchored object to measure YDistance to.

See Also | [Example](#)

**Signature**

object.YPositionTo

object

[AecAnchorEntToCurve](#) The object this property applies to.

YPositionTo

[AecPosition](#) enum; read-write

The position that [YDistance](#) measures to.
YSpacing Property

Specifies the spacing between nodes along the Y direction of the layout grid.

See Also | Example

Signature

```
object.YSpacing
```

object

```
AecLayoutGrid2D, AecLayoutGrid3D
```

The object this property applies to.

YSpacing

```
ACAD_DISTANCE; read-write
```

The spacing between nodes along the Y direction of the layout grid.
ZCount Property

Specifies the number of nodes along the Z direction.

See Also | Example

Signature

object.ZCount

object

AecLayoutGrid2D, AecLayoutGrid3D
The object this property applies to.

ZCount

Long; read-write
The number of nodes along the Z direction.
ZDistance Property

Specifies the distance above the curve to the anchored object.

See Also | Example

Signature

object.ZDistance

object

AecAnchorEntToCurve The object this property applies to.

ZDistance

Double; read-write
Distance above the curve to the anchored object.
ZOffset Property

Specifies the offset in the Z direction of the view block within the multi-view block.

See Also | Example

Signature

object.ZOffset

object

AecViewBlock The object this property applies to.

ZOffset

Double; read-write
The offset in the Z direction of the view block within the multi-view block.
**Zoom Property**

Specifies the focal length of the camera, in millimeters.

See Also | Example

**Signature**

object.Zoom

object

AecCamera The object this property applies to.

Zoom

Double; read-write
Length of the lens in millimeters.

**Remarks**

The LENSLENGTH system variable will be reset to the Zoom value when the camera generates a view.
ZPositionFrom Property

Specifies the position on the curve from which the ZDistance to an anchored object is measured.

See Also | Example

Signature

object.ZPositionFrom

object

AecAnchorEntToCurve The object this property applies to.

ZPositionFrom

AecCurveHeightPosition enum; read-write
The position on the curve from which ZDistance is measured.
ZPositionTo Property

Specifies the position on the anchored object to measure ZDistance to.

See Also | Example

Signature

object.ZPositionTo

object

AecAnchorEntToCurve The object this property applies to.

ZPositionTo

AecHeight enum; read-write
The position that ZDistance measures to.
ZRotation Property

Specifies the rotation of the anchored object around the Z axis.

See Also | Example

Signature

object.ZRotation

object

AecAnchorEntToLayoutCell, AecAnchorEntToLayoutNode, AecAnchorEntToLayoutVolume, AecAnchorEntToCurve, AecAnchorLeadEntToNode

The object this property applies to.

ZRotation

Double; read-write

Rotation of the anchored object around the Z axis.
ZSpacing Property

Specifies the spacing between nodes along the Z direction of the layout grid.

See Also | Example

Signature

object.ZSpacing

object

AecLayoutGrid2D, AecLayoutGrid3D
The object this property applies to.

ZSpacing

ACAD_DISTANCE; read-write
The spacing between nodes along the Z direction of the layout grid.
Modified Event

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

See Also | Example

Signature

object.Modified(Entity)

Aec2dSection, AecAnchor, AecAnchorEntToCurve, AecAnchorEntToGridAssembly, AecAnchorEntToLayoutCell, AecAnchorEntToLayoutNode, AecAnchorEntToLayoutVolume, AecAnchorLeadEntToNode, AecAnchorToRef, AecBlockRef, AecCamera, AecCellLayoutTool, AecClipVol, AecClipVolRes, AecDictionary, AecDictRecord, AecEditInPlaceProfile, AecEntity, AecEntRef, AecGeo, AecGridAssembly, AecLayerKeyStyle, AecLayerKeyStyles, AecLayoutCurve, AecLayoutGrid2D, AecLayoutGrid3D, AecLayoutTool, AecMaskBlockRef, AecMaskBlockStyle, AecMaskBlockStyles, AecMassElement, AecMassElementStyle, AecMassElementStyles, AecMassGroup, AecMVBlockRef, AecMVBlockStyle, AecMVBlockStyles, AecObject, AecPolygon, AecPolygonStyle, AecPolygonStyles, AecProfileStyle, AecProfileStyles, AecSlice, AecVolumeLayoutTool

An object expression that evaluates to a valid container object.
**Entity**

**A Drawing Object**
The object in the drawing that is modified can be any one of the drawing objects.

**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.
AutoCAD object and layer colors.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>acByBlock</td>
<td>0</td>
<td>The color defined for the object's block.</td>
</tr>
<tr>
<td>acRed</td>
<td>1</td>
<td>(red)</td>
</tr>
<tr>
<td>acYellow</td>
<td>2</td>
<td>(yellow)</td>
</tr>
<tr>
<td>acGreen</td>
<td>3</td>
<td>(green)</td>
</tr>
<tr>
<td>acCyan</td>
<td>4</td>
<td>(cyan)</td>
</tr>
<tr>
<td>acBlue</td>
<td>5</td>
<td>(blue)</td>
</tr>
<tr>
<td>acMagenta</td>
<td>6</td>
<td>(magenta)</td>
</tr>
<tr>
<td>acWhite</td>
<td>7</td>
<td>(white)</td>
</tr>
<tr>
<td>acByLayer (not valid for Layer object)</td>
<td>256</td>
<td>The color of the layer the object is on.</td>
</tr>
</tbody>
</table>
colors.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The color defined for the object's block.</td>
</tr>
<tr>
<td>1</td>
<td>• (red)</td>
</tr>
<tr>
<td>2</td>
<td>• (yellow)</td>
</tr>
<tr>
<td>3</td>
<td>• (green)</td>
</tr>
<tr>
<td>4</td>
<td>• (cyan)</td>
</tr>
<tr>
<td>5</td>
<td>• (blue)</td>
</tr>
<tr>
<td>6</td>
<td>• (magenta)</td>
</tr>
<tr>
<td>7</td>
<td>◊ (white)</td>
</tr>
<tr>
<td>256</td>
<td>The color of the layer the object is on.</td>
</tr>
</tbody>
</table>
Lineweights.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>acLnWtByBlock</td>
<td>-2</td>
</tr>
<tr>
<td>acLnWtByLayer</td>
<td>-1</td>
</tr>
<tr>
<td>acLnWtByLwDefault</td>
<td>-3</td>
</tr>
<tr>
<td>acLnWt000</td>
<td>0</td>
</tr>
<tr>
<td>acLnWt005</td>
<td>5</td>
</tr>
<tr>
<td>acLnWt009</td>
<td>9</td>
</tr>
<tr>
<td>acLnWt013</td>
<td>13</td>
</tr>
<tr>
<td>acLnWt015</td>
<td>15</td>
</tr>
<tr>
<td>acLnWt018</td>
<td>18</td>
</tr>
<tr>
<td>acLnWt020</td>
<td>20</td>
</tr>
<tr>
<td>acLnWt025</td>
<td>25</td>
</tr>
<tr>
<td>acLnWt030</td>
<td>30</td>
</tr>
<tr>
<td>acLnWt035</td>
<td>35</td>
</tr>
<tr>
<td>acLnWt040</td>
<td>40</td>
</tr>
<tr>
<td>acLnWt050</td>
<td>50</td>
</tr>
<tr>
<td>acLnWt053</td>
<td>53</td>
</tr>
<tr>
<td>acLnWt060</td>
<td>60</td>
</tr>
<tr>
<td>acLnWt070</td>
<td>70</td>
</tr>
<tr>
<td>acLnWt080</td>
<td>80</td>
</tr>
<tr>
<td>acLnWt090</td>
<td>90</td>
</tr>
<tr>
<td>acLnWt100</td>
<td>100</td>
</tr>
<tr>
<td>acLnWt106</td>
<td>106</td>
</tr>
<tr>
<td>acLnWt120</td>
<td>120</td>
</tr>
<tr>
<td>acLnWt140</td>
<td>140</td>
</tr>
<tr>
<td>acLnWt158</td>
<td>158</td>
</tr>
<tr>
<td>acLnWt200</td>
<td>200</td>
</tr>
<tr>
<td>acLnWt211</td>
<td>211</td>
</tr>
</tbody>
</table>
Drawing units.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>acEnglish</td>
<td>English (Imperial) units</td>
</tr>
<tr>
<td>acMetric</td>
<td>Metric units</td>
</tr>
</tbody>
</table>
The state of the application window.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>acNorm</td>
<td>1</td>
<td>Normal (neither minimized or maximized)</td>
</tr>
<tr>
<td>acMin</td>
<td>2</td>
<td>Minimized</td>
</tr>
<tr>
<td>acMax</td>
<td>3</td>
<td>Maximized</td>
</tr>
</tbody>
</table>
Unit of measurement.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aecUnitAcre</td>
<td>57</td>
<td>Acre</td>
</tr>
<tr>
<td>aecUnitAmpere</td>
<td>5</td>
<td>Ampere</td>
</tr>
<tr>
<td>aecUnitAngstrom</td>
<td>17</td>
<td>Angstrom</td>
</tr>
<tr>
<td>aecUnitAre</td>
<td>58</td>
<td>Are (100 square meters)</td>
</tr>
<tr>
<td>aecUnitAstronomicalUnit</td>
<td>18</td>
<td>Astronomical Unit</td>
</tr>
<tr>
<td>aecUnitBarn</td>
<td>59</td>
<td>Barn</td>
</tr>
<tr>
<td>aecUnitBarrel</td>
<td>98</td>
<td>Barrel</td>
</tr>
<tr>
<td>aecUnitBoardFoot</td>
<td>99</td>
<td>Board Foot</td>
</tr>
<tr>
<td>aecUnitBolt</td>
<td>36</td>
<td>Bolt (40 yards)</td>
</tr>
<tr>
<td>aecUnitBushel</td>
<td>100</td>
<td>Bushel</td>
</tr>
<tr>
<td>aecUnitCable</td>
<td>35</td>
<td>Cable (720 feet)</td>
</tr>
<tr>
<td>aecUnitCaliber</td>
<td>32</td>
<td>Caliber (.01 inches)</td>
</tr>
<tr>
<td>aecUnitCandela</td>
<td>7</td>
<td>Candela</td>
</tr>
<tr>
<td>aecUnitCentare</td>
<td>60</td>
<td>.01 are (1 square meter)</td>
</tr>
<tr>
<td>aecUnitCentiliter</td>
<td>101</td>
<td>Centiliter</td>
</tr>
<tr>
<td>aecUnitCentimeter</td>
<td>24</td>
<td>Centimeter</td>
</tr>
<tr>
<td>aecUnitCentury</td>
<td>175</td>
<td>Century</td>
</tr>
<tr>
<td>aecUnitChain</td>
<td>37</td>
<td>Chain</td>
</tr>
<tr>
<td>aecUnitChainEngineers</td>
<td>38</td>
<td>100 feet (30.48 meters)</td>
</tr>
<tr>
<td>aecUnitCord</td>
<td>102</td>
<td>Cord</td>
</tr>
<tr>
<td>aecUnitCubicAngstrom</td>
<td>122</td>
<td>Cubic Angstrom</td>
</tr>
<tr>
<td>aecUnitCubicAstronomicalUnit</td>
<td>123</td>
<td>Cubic Astronomical Unit</td>
</tr>
<tr>
<td>Unit</td>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Cubic Bolt</td>
<td>124</td>
<td>Cubic Bolt</td>
</tr>
<tr>
<td>Cubic Cable</td>
<td>125</td>
<td>Cubic Cable</td>
</tr>
<tr>
<td>Cubic Caliber</td>
<td>126</td>
<td>Cubic Caliber</td>
</tr>
<tr>
<td>Cubic Centimeter</td>
<td>103</td>
<td>Cubic Centimeter</td>
</tr>
<tr>
<td>Cubic Chain</td>
<td>127</td>
<td>Cubic Chain</td>
</tr>
<tr>
<td>Cubic Cubit</td>
<td>128</td>
<td>Cubic Cubit</td>
</tr>
<tr>
<td>Cubic Decimeter</td>
<td>177</td>
<td>Cubic Decimeter</td>
</tr>
<tr>
<td>Cubic Fathom</td>
<td>129</td>
<td>Cubic Fathom</td>
</tr>
<tr>
<td>Cubic Foot</td>
<td>130</td>
<td>Cubic Foot</td>
</tr>
<tr>
<td>Cubic Furlong</td>
<td>131</td>
<td>Cubic Furlong</td>
</tr>
<tr>
<td>Cubic Gigameter</td>
<td>132</td>
<td>Cubic Gigameter</td>
</tr>
<tr>
<td>Cubic Hand</td>
<td>133</td>
<td>Cubic Hand</td>
</tr>
<tr>
<td>Cubic Hectometer</td>
<td>134</td>
<td>Cubic Hectometer</td>
</tr>
<tr>
<td>Cubic Inch</td>
<td>135</td>
<td>Cubic Inch</td>
</tr>
<tr>
<td>Cubic League Nautical</td>
<td>136</td>
<td>Cubic League Nautical</td>
</tr>
<tr>
<td>Cubic League Statute</td>
<td>137</td>
<td>Cubic League Statute</td>
</tr>
<tr>
<td>Cubic Light Year</td>
<td>138</td>
<td>Cubic Light Year</td>
</tr>
<tr>
<td>Cubic Link</td>
<td>139</td>
<td>Cubic Link</td>
</tr>
<tr>
<td>Cubic Meter</td>
<td>96</td>
<td>Cubic Meter</td>
</tr>
<tr>
<td>Cubic Mil</td>
<td>140</td>
<td>Cubic Mil</td>
</tr>
<tr>
<td>Cubic Mile Nautical</td>
<td>141</td>
<td>Cubic Mile Nautical</td>
</tr>
<tr>
<td>Cubic Mile Statute</td>
<td>142</td>
<td>Cubic Mile Statute</td>
</tr>
<tr>
<td>Cubic Millimeter</td>
<td>143</td>
<td>Cubic Millimeter</td>
</tr>
<tr>
<td>Cubic Millimicron</td>
<td>144</td>
<td>Cubic Millimicron</td>
</tr>
<tr>
<td>Cubic Nautical League</td>
<td>136</td>
<td>Cubic Nautical League</td>
</tr>
<tr>
<td>Cubic Nautical Mile</td>
<td>141</td>
<td>Cubic Nautical Mile</td>
</tr>
<tr>
<td>Cubic Pace</td>
<td>145</td>
<td>Cubic Pace</td>
</tr>
<tr>
<td>Cubic Palm</td>
<td>146</td>
<td>Cubic Palm</td>
</tr>
<tr>
<td>Cubic Parsec</td>
<td>147</td>
<td>Cubic Parsec</td>
</tr>
<tr>
<td>aecUnitCubicPerch</td>
<td>148</td>
<td>Cubic Perch</td>
</tr>
<tr>
<td>aecUnitCubicRope</td>
<td>149</td>
<td>Cubic Rope</td>
</tr>
<tr>
<td>aecUnitCubicSkein</td>
<td>150</td>
<td>Cubic Skein</td>
</tr>
<tr>
<td>aecUnitCubicSpan</td>
<td>151</td>
<td>Cubic Span</td>
</tr>
<tr>
<td>aecUnitCubicLeagueStatute</td>
<td>137</td>
<td>Cubic Statute League</td>
</tr>
<tr>
<td>aecUnitCubicMileStatute</td>
<td>142</td>
<td>Cubic Statute Mile</td>
</tr>
<tr>
<td>aecUnitCubicSurveyFoot</td>
<td>152</td>
<td>Cubic Survey Foot</td>
</tr>
<tr>
<td>aecUnitCubicYard</td>
<td>153</td>
<td>Cubic Yard</td>
</tr>
<tr>
<td>aecUnitCubit</td>
<td>40</td>
<td>Cubit</td>
</tr>
<tr>
<td>aecUnitDalton</td>
<td>154</td>
<td>Dalton (atomic mass unit)</td>
</tr>
<tr>
<td>aecUnitDay</td>
<td>169</td>
<td>Day</td>
</tr>
<tr>
<td>aecUnitDecade</td>
<td>174</td>
<td>Decade</td>
</tr>
<tr>
<td>aecUnitDecimeter</td>
<td>23</td>
<td>Decimeter</td>
</tr>
<tr>
<td>aecUnitDecistere</td>
<td>104</td>
<td>Decistere</td>
</tr>
<tr>
<td>aecUnitDekaliter</td>
<td>105</td>
<td>Dekaliter</td>
</tr>
<tr>
<td>aecUnitDekameter</td>
<td>22</td>
<td>Dekameter</td>
</tr>
<tr>
<td>aecUnitDekastere</td>
<td>106</td>
<td>Dekastere</td>
</tr>
<tr>
<td>aecUnitDimensionless</td>
<td>0</td>
<td>Dimensionless</td>
</tr>
<tr>
<td>aecUnitDram</td>
<td>107</td>
<td>Dram</td>
</tr>
<tr>
<td>aecUnitDryPint</td>
<td>108</td>
<td>Dry Pint</td>
</tr>
<tr>
<td>aecUnitDryQuart</td>
<td>109</td>
<td>Dry Quart</td>
</tr>
<tr>
<td>aecUnitFathoms</td>
<td>34</td>
<td>Fathom</td>
</tr>
<tr>
<td>aecUnitFirkin</td>
<td>110</td>
<td>Firkin</td>
</tr>
<tr>
<td>aecUnitFluidOunce</td>
<td>116</td>
<td>Fluid Ounce</td>
</tr>
<tr>
<td>aecUnitFluidPint</td>
<td>118</td>
<td>Fluid Pint</td>
</tr>
<tr>
<td>aecUnitFluidQuart</td>
<td>119</td>
<td>Fluid Quart</td>
</tr>
<tr>
<td>aecUnitFoot</td>
<td>30</td>
<td>Foot</td>
</tr>
<tr>
<td>aecUnitFortnight</td>
<td>171</td>
<td>Fortnight</td>
</tr>
<tr>
<td>aecUnitFurlong</td>
<td>41</td>
<td>Furlong</td>
</tr>
<tr>
<td>Unit</td>
<td>Conversion Factor</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>Gallon</td>
<td>111</td>
<td>Gallon</td>
</tr>
<tr>
<td>Gigameter</td>
<td>19</td>
<td>Gigameter</td>
</tr>
<tr>
<td>Gill (.5 cups)</td>
<td>112</td>
<td>Gill</td>
</tr>
<tr>
<td>Grain</td>
<td>155</td>
<td>Grain</td>
</tr>
<tr>
<td>Gram</td>
<td>156</td>
<td>Gram</td>
</tr>
<tr>
<td>Hand (4 inches)</td>
<td>42</td>
<td>Hand</td>
</tr>
<tr>
<td>Hectare</td>
<td>61</td>
<td>Hectare</td>
</tr>
<tr>
<td>Hectometer</td>
<td>21</td>
<td>Hectometer</td>
</tr>
<tr>
<td>Hogshead</td>
<td>113</td>
<td>Hogshead</td>
</tr>
<tr>
<td>Hour</td>
<td>168</td>
<td>Hour</td>
</tr>
<tr>
<td>Hundredweight</td>
<td>157</td>
<td>Hundredweight</td>
</tr>
<tr>
<td>Inch</td>
<td>31</td>
<td>Inch</td>
</tr>
<tr>
<td>Joule</td>
<td>12</td>
<td>Joule</td>
</tr>
<tr>
<td>Kelvin</td>
<td>4</td>
<td>Kelvin</td>
</tr>
<tr>
<td>Kilogram</td>
<td>1</td>
<td>Kilogram</td>
</tr>
<tr>
<td>Kiloliter</td>
<td>114</td>
<td>Kiloliter</td>
</tr>
<tr>
<td>Kilometer</td>
<td>20</td>
<td>Kilometer</td>
</tr>
<tr>
<td>League Nautical</td>
<td>45</td>
<td>League Nautical</td>
</tr>
<tr>
<td>League Statute</td>
<td>46</td>
<td>League Statute</td>
</tr>
<tr>
<td>Light Year</td>
<td>28</td>
<td>Light Year</td>
</tr>
<tr>
<td>Link (7.92 inches)</td>
<td>39</td>
<td>Link</td>
</tr>
<tr>
<td>Liter</td>
<td>97</td>
<td>Liter</td>
</tr>
<tr>
<td>Long Ton</td>
<td>158</td>
<td>2240 pounds (1.016 metric tons)</td>
</tr>
<tr>
<td>Lumen</td>
<td>11</td>
<td>Lumen</td>
</tr>
<tr>
<td>Lux</td>
<td>16</td>
<td>Lux</td>
</tr>
<tr>
<td>Meter</td>
<td>2</td>
<td>Meter</td>
</tr>
<tr>
<td>Micron</td>
<td>26</td>
<td>Micron</td>
</tr>
<tr>
<td>Mil</td>
<td>47</td>
<td>Mil</td>
</tr>
<tr>
<td>Unit</td>
<td>Value</td>
<td>Definition</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>aecUnitMile</td>
<td>44</td>
<td>Mile</td>
</tr>
<tr>
<td>aecUnitMileNautical</td>
<td>43</td>
<td>Mile Nautical</td>
</tr>
<tr>
<td>aecUnitMillenium</td>
<td>176</td>
<td>Millenium</td>
</tr>
<tr>
<td>aecUnitMillimeter</td>
<td>25</td>
<td>Millimeter</td>
</tr>
<tr>
<td>aecUnitMillimicron</td>
<td>27</td>
<td>Millimicron</td>
</tr>
<tr>
<td>aecUnitMinim</td>
<td>115</td>
<td>Minim</td>
</tr>
<tr>
<td>aecUnitMinute</td>
<td>167</td>
<td>Minute</td>
</tr>
<tr>
<td>aecUnitMole</td>
<td>6</td>
<td>Mole</td>
</tr>
<tr>
<td>aecUnitLeagueNautical</td>
<td>45</td>
<td>Nautical League</td>
</tr>
<tr>
<td>aecUnitTonne</td>
<td>166</td>
<td>Metric Ton (tonne)</td>
</tr>
<tr>
<td>aecUnitMileNautical</td>
<td>43</td>
<td>Nautical Mile</td>
</tr>
<tr>
<td>aecUnitNewton</td>
<td>10</td>
<td>Newton</td>
</tr>
<tr>
<td>aecUnitOunce</td>
<td>159</td>
<td>Ounce</td>
</tr>
<tr>
<td>aecUnitOunceTroy</td>
<td>160</td>
<td>Ounce Troy</td>
</tr>
<tr>
<td>aecUnitPace</td>
<td>48</td>
<td>Pace</td>
</tr>
<tr>
<td>aecUnitPalm</td>
<td>49</td>
<td>Palm (3 inches)</td>
</tr>
<tr>
<td>aecUnitParsec</td>
<td>29</td>
<td>Parsec</td>
</tr>
<tr>
<td>aecUnitPascals</td>
<td>13</td>
<td>Pascal</td>
</tr>
<tr>
<td>aecUnitPeck</td>
<td>117</td>
<td>Peck</td>
</tr>
<tr>
<td>aecUnitPennyweight</td>
<td>161</td>
<td>Pennyweight</td>
</tr>
<tr>
<td>aecUnitPound</td>
<td>162</td>
<td>Pound</td>
</tr>
<tr>
<td>aecUnitRadian</td>
<td>8</td>
<td>Radian</td>
</tr>
<tr>
<td>aecUnitRegisterTon</td>
<td>120</td>
<td>Register Ton</td>
</tr>
<tr>
<td>aecUnitRod</td>
<td>50</td>
<td>Rod</td>
</tr>
<tr>
<td>aecUnitRope</td>
<td>51</td>
<td>rope</td>
</tr>
<tr>
<td>aecUnitSecond</td>
<td>3</td>
<td>Second</td>
</tr>
<tr>
<td>aecUnitSiderealYear</td>
<td>172</td>
<td>Sidereal Year</td>
</tr>
<tr>
<td>aecUnitSkein</td>
<td>52</td>
<td>Skein</td>
</tr>
<tr>
<td>aecUnitSlug</td>
<td>163</td>
<td>Slug</td>
</tr>
<tr>
<td>------------</td>
<td>-----</td>
<td>------</td>
</tr>
<tr>
<td>aecUnitSpan</td>
<td>53</td>
<td>Span</td>
</tr>
<tr>
<td>aecUnitSquareAngstrom</td>
<td>62</td>
<td>Square Angstrom</td>
</tr>
<tr>
<td>aecUnitSquareAstronomicalUnit</td>
<td>63</td>
<td>Square Angstrom</td>
</tr>
<tr>
<td>aecUnitSquareBolt</td>
<td>64</td>
<td>Square Bolt</td>
</tr>
<tr>
<td>aecUnitSquareCable</td>
<td>65</td>
<td>Square Cable</td>
</tr>
<tr>
<td>aecUnitSquareCaliber</td>
<td>66</td>
<td>Square Caliber</td>
</tr>
<tr>
<td>aecUnitSquareCentimeter</td>
<td>67</td>
<td>Square Centimeter</td>
</tr>
<tr>
<td>aecUnitSquareChain</td>
<td>94</td>
<td>Square Chain</td>
</tr>
<tr>
<td>aecUnitSquareCubit</td>
<td>68</td>
<td>Square Cubit</td>
</tr>
<tr>
<td>aecUnitSquareDecimeter</td>
<td>69</td>
<td>Square Decimeter</td>
</tr>
<tr>
<td>aecUnitSquareFathom</td>
<td>70</td>
<td>Square Fathom</td>
</tr>
<tr>
<td>aecUnitSquareFoot</td>
<td>56</td>
<td>Square Foot</td>
</tr>
<tr>
<td>aecUnitSquareFurlong</td>
<td>71</td>
<td>Square Furlong</td>
</tr>
<tr>
<td>aecUnitSquareGigameter</td>
<td>72</td>
<td>Square Gigameter</td>
</tr>
<tr>
<td>aecUnitSquareHand</td>
<td>73</td>
<td>Square Hand</td>
</tr>
<tr>
<td>aecUnitSquareInch</td>
<td>74</td>
<td>Square Inch</td>
</tr>
<tr>
<td>aecUnitSquareKilometer</td>
<td>75</td>
<td>Square Kilometer</td>
</tr>
<tr>
<td>aecUnitSquareLeagueNautical</td>
<td>76</td>
<td>Square League Nautical</td>
</tr>
<tr>
<td>aecUnitSquareLeagueStatute</td>
<td>77</td>
<td>Square League Statute</td>
</tr>
<tr>
<td>aecUnitSquareLightYear</td>
<td>78</td>
<td>Square Light Year</td>
</tr>
<tr>
<td>aecUnitSquareLink</td>
<td>79</td>
<td>Square Link</td>
</tr>
<tr>
<td>aecUnitSquareMeters</td>
<td>55</td>
<td>Square Meter</td>
</tr>
<tr>
<td>aecUnitSquareMicron</td>
<td>80</td>
<td>Square Micron</td>
</tr>
<tr>
<td>aecUnitSquareMil</td>
<td>81</td>
<td>Square Mil</td>
</tr>
<tr>
<td>aecUnitSquareMileNautical</td>
<td>82</td>
<td>Square Mile Nautical</td>
</tr>
<tr>
<td>aecUnitSquareMileStatute</td>
<td>83</td>
<td>Square Mile Statute</td>
</tr>
<tr>
<td>aecUnitSquareMillimicron</td>
<td>84</td>
<td>Square Millimicron</td>
</tr>
<tr>
<td>Unit</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>aecUnitSquareLeagueNautical</td>
<td>76</td>
<td>Square Nautical League</td>
</tr>
<tr>
<td>aecUnitSquareMileNautical</td>
<td>82</td>
<td>Square Nautical Mile</td>
</tr>
<tr>
<td>aecUnitSquarePace</td>
<td>85</td>
<td>Square Pace</td>
</tr>
<tr>
<td>aecUnitSquarePalm</td>
<td>86</td>
<td>Square Palm</td>
</tr>
<tr>
<td>aecUnitSquareParsec</td>
<td>87</td>
<td>Square Parsec</td>
</tr>
<tr>
<td>aecUnitSquarePerch</td>
<td>88</td>
<td>Square Perch</td>
</tr>
<tr>
<td>aecUnitSquareRope</td>
<td>89</td>
<td>Square Rope</td>
</tr>
<tr>
<td>aecUnitSquareSkein</td>
<td>90</td>
<td>Square Skein</td>
</tr>
<tr>
<td>aecUnitSquareSpan</td>
<td>91</td>
<td>Square Span</td>
</tr>
<tr>
<td>aecUnitSquareLeagueStatute</td>
<td>77</td>
<td>Square Statute League</td>
</tr>
<tr>
<td>aecUnitSquareMileStatute</td>
<td>83</td>
<td>Square Statute Mile</td>
</tr>
<tr>
<td>aecUnitSquareSurveyFoot</td>
<td>92</td>
<td>Square Survey Foot</td>
</tr>
<tr>
<td>aecUnitSquareYard</td>
<td>93</td>
<td>Square Yard</td>
</tr>
<tr>
<td>aecUnitLeagueStatute</td>
<td>46</td>
<td>Statute League</td>
</tr>
<tr>
<td>aecUnitSteradian</td>
<td>9</td>
<td>Steradian</td>
</tr>
<tr>
<td>aecUnitStone</td>
<td>164</td>
<td>Stone</td>
</tr>
<tr>
<td>aecUnitSurveyFoot</td>
<td>54</td>
<td>Survey Foot</td>
</tr>
<tr>
<td>aecUnitTon</td>
<td>165</td>
<td>Ton</td>
</tr>
<tr>
<td>aecUnitTonne</td>
<td>166</td>
<td>Tonne (metric ton)</td>
</tr>
<tr>
<td>aecUnitTownship</td>
<td>95</td>
<td>Township (36 square miles)</td>
</tr>
<tr>
<td>aecUnitTropicalYear</td>
<td>173</td>
<td>365 days, 5 hours, 48 minutes, 45.51 seconds (astronomical year)</td>
</tr>
<tr>
<td>aecUnitTun</td>
<td>121</td>
<td>Tun</td>
</tr>
<tr>
<td>aecUnitVolt</td>
<td>15</td>
<td>Volt</td>
</tr>
<tr>
<td>aecUnitWatt</td>
<td>14</td>
<td>Watt</td>
</tr>
<tr>
<td>aecUnitWeek</td>
<td>170</td>
<td>Week</td>
</tr>
<tr>
<td>aecUnitYards</td>
<td>33</td>
<td>Yard</td>
</tr>
</tbody>
</table>
Position on the curve.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aecCurveHeightPositionBottom</td>
<td>0</td>
<td>Bottom</td>
</tr>
<tr>
<td>aecCurveHeightPositionCenter</td>
<td>1</td>
<td>Center</td>
</tr>
<tr>
<td>aecCurveHeightPositionTop</td>
<td>2</td>
<td>Top</td>
</tr>
</tbody>
</table>
Position on the curve.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aecCurvePositionStart</td>
<td>0</td>
<td>Start</td>
</tr>
<tr>
<td>aecCurvePositionMiddle</td>
<td>1</td>
<td>Middle</td>
</tr>
<tr>
<td>aecCurvePositionEnd</td>
<td>2</td>
<td>End</td>
</tr>
</tbody>
</table>
Position on the curve.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aecCurveWidthPositionLeft</td>
<td>0</td>
<td>Left</td>
</tr>
<tr>
<td>aecCurveWidthPositionCenter</td>
<td>1</td>
<td>Center</td>
</tr>
<tr>
<td>aecCurveWidthPositionRight</td>
<td>2</td>
<td>Right</td>
</tr>
</tbody>
</table>
Position measured to.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aecEdgePositionStart</td>
<td>0</td>
<td>Start</td>
</tr>
<tr>
<td>aecEdgePositionCenter</td>
<td>1</td>
<td>Center</td>
</tr>
<tr>
<td>aecEdgePositionEnd</td>
<td>2</td>
<td>End</td>
</tr>
</tbody>
</table>
Position on the object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aecHeightBottom</td>
<td>0</td>
<td>Bottom</td>
</tr>
<tr>
<td>aecHeightCenter</td>
<td>1</td>
<td>Center</td>
</tr>
<tr>
<td>aecHeightTop</td>
<td>2</td>
<td>Top</td>
</tr>
</tbody>
</table>
Alignment in relation to baseline.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aecInfillAlignCentered</td>
<td>0</td>
<td>Centered</td>
</tr>
<tr>
<td>aecInfillAlignFrontFlush</td>
<td>1</td>
<td>In front of the baseline</td>
</tr>
<tr>
<td>aecInfillAlignBackFlush</td>
<td>2</td>
<td>In back of the baseline</td>
</tr>
</tbody>
</table>
The shape of a layout grid.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aecLayoutShapeRectangular</td>
<td>0</td>
<td>Rectangular layout</td>
</tr>
<tr>
<td>aecLayoutShapeRadial</td>
<td>1</td>
<td>Radial layout</td>
</tr>
<tr>
<td>aecLayoutShapeCustom</td>
<td>2</td>
<td>Custom layout</td>
</tr>
</tbody>
</table>
Type of node spacing along the curve.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aecLayoutTypeManualSpacing</td>
<td>0</td>
<td>Manual spacing</td>
</tr>
<tr>
<td>aecLayoutTypeAutoSpacingEven</td>
<td>1</td>
<td>Spaced evenly along the curve</td>
</tr>
<tr>
<td>aecLayoutTypeAutoSpacingBay</td>
<td>2</td>
<td>Automatic spacing, determined by the specified distance between nodes</td>
</tr>
</tbody>
</table>
Subtype of the mass element.

<table>
<thead>
<tr>
<th><strong>AecMassElementSubType</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td>aecMassElementSubTypeUnspecified</td>
</tr>
<tr>
<td>aecMassElementSubTypeProfileInternal</td>
</tr>
<tr>
<td>aecMassElementSubTypeProfileExternal</td>
</tr>
</tbody>
</table>
Shape of the mass element.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aecMassElementTypeArch</td>
<td>1</td>
<td>Arch</td>
</tr>
<tr>
<td>aecMassElementTypeBarrelVault</td>
<td>2</td>
<td>Barrel vault</td>
</tr>
<tr>
<td>aecMassElementTypeBox</td>
<td>0</td>
<td>Box</td>
</tr>
<tr>
<td>aecMassElementTypeDoric</td>
<td>3</td>
<td>Simple column</td>
</tr>
<tr>
<td>aecMassElementTypeCone</td>
<td>4</td>
<td>Cone</td>
</tr>
<tr>
<td>aecMassElementTypeCylinder</td>
<td>5</td>
<td>Cylinder</td>
</tr>
<tr>
<td>aecMassElementTypeDome</td>
<td>6</td>
<td>Dome</td>
</tr>
<tr>
<td>aecMassElementTypeExtrusion</td>
<td>12</td>
<td>Extrusion of a profile</td>
</tr>
<tr>
<td>aecMassElementTypeFreeForm</td>
<td>14</td>
<td>Free form</td>
</tr>
<tr>
<td>aecMassElementTypeGable</td>
<td>7</td>
<td>Gable</td>
</tr>
<tr>
<td>aecMassElementTypeIsoscelesTriangle</td>
<td>8</td>
<td>Isosceles triangle</td>
</tr>
<tr>
<td>aecMassElementTypePyramid</td>
<td>10</td>
<td>Pyramid</td>
</tr>
<tr>
<td>aecMassElementTypeRevolution</td>
<td>13</td>
<td>Revolved profile</td>
</tr>
<tr>
<td>aecMassElementTypeRightTriangle</td>
<td>9</td>
<td>Right triangle</td>
</tr>
<tr>
<td>aecMassElementTypeSphere</td>
<td>11</td>
<td>Sphere</td>
</tr>
</tbody>
</table>
Type of operation combining objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aecMassOperationAdd</td>
<td>0</td>
<td>Addition</td>
</tr>
<tr>
<td>aecMassOperationSubtract</td>
<td>1</td>
<td>Subtraction</td>
</tr>
<tr>
<td>aecMassOperationIntersect</td>
<td>2</td>
<td>Intersection</td>
</tr>
</tbody>
</table>
Placement of edges on AEC Polygon vertices.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aecPolygonEdgeWidthJustificationIn</td>
<td>0</td>
<td>Inside edge</td>
</tr>
<tr>
<td>aecPolygonEdgeWidthJustificationCenter</td>
<td>1</td>
<td>Center</td>
</tr>
<tr>
<td>aecPolygonEdgeWidthJustificationOut</td>
<td>2</td>
<td>Outside edge</td>
</tr>
</tbody>
</table>
Position on the object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aecPositionFront</td>
<td>0</td>
<td>Front</td>
</tr>
<tr>
<td>aecPositionCenter</td>
<td>1</td>
<td>Center</td>
</tr>
<tr>
<td>aecPositionBack</td>
<td>2</td>
<td>Back</td>
</tr>
</tbody>
</table>
Sub Example_Add_AecProfile()

' This example defines a custom profile consisting of
' of two triangles, one within the other.

Dim doc As AecBaseDocument
Dim app As New AecBaseApplication
Dim profileStyle As AecProfileStyle
Dim profile As New AecProfile
Dim profileName As String

app.Init ThisDrawing.Application
Set doc = app.ActiveDocument

' Get existing or add a new profile style.
On Error Resume Next
profileName = "BOOBYPRIZE"
Set profileStyle = doc.ProfileStyles.Add(profileName)
If Err.Number = 0 Then
    Err.Clear
    On Error GoTo 0
    Set profileStyle = doc.ProfileStyles.Item(profileName)
End If

' Define definition points for rings.
Dim OuterPts(0 To 7) As Double
Dim InnerPts(0 To 7) As Double

OuterPts(0) = 0: OuterPts(1) = 0
OuterPts(2) = 648: OuterPts(3) = 0
OuterPts(4) = 324: OuterPts(5) = 324
OuterPts(6) = 0: OuterPts(7) = 0

InnerPts(0) = 120: InnerPts(1) = 60
InnerPts(2) = 528: InnerPts(3) = 60
InnerPts(4) = 324: InnerPts(5) = 264
InnerPts(6) = 120: InnerPts(7) = 60

' Create and set the outer ring.
Dim ring1 As AecRing
Set ring1 = profile.Rings.Add
ring1.FromPoints (OuterPts)
ring1.Void = False

' Create and set the inner ring.
Dim ring2 As AecRing
Set ring2 = profile.Rings.Add
ring2.FromPoints (InnerPts)
ring2.Void = True

' Set the profile definition.
Set profileStyle.profile = profile

MsgBox ("Profile """" & profileName & """" created.")

End Sub
**AdjustSizing Example**

Sub Example_AdjustSizing()

' This example modifies the top offset of the selected object
' in relation to its anchor point on the grid assembly. It
' prompts the user to select the object to be modified,
' and then it prompts the user to specify the number of inches
' to adjust the offset by. The AdjustSizing property is checked,
' and if it is not currently set to True, it is reset and the
' the user is advised of the change.

' Use this example with a drawing that contains a window
' assembly and one or more AEC objects attached to the
' assembly.

Dim ent As AcadEntity
Dim geo As AecGeo
Dim anchor As AecAnchor

Dim offset As String
Dim offset_adjust As Double

On Error Resume Next     ' Handle errors in code.

' Prompt user to select an object.
ThisDrawing.Utility.GetEntity ent, pt, "Select object anchored to window asse

' Make sure user selected an AEC object, and that the object
' is anchored to a grid assembly.
If ent Is Nothing Then
    MsgBox "Nothing was selected.", vbExclamation, "AdjustSizing Example"
ElseIf TypeOf ent Is AecGeo Then
    Set geo = ent

    ' Get the anchor the object is attached to.
    Set anchor = geo.GetAnchor
On Error GoTo 0
If anchor Is Nothing Then
    MsgBox "Selected object is not anchored.", vbExclamation, "AdjustSizing Example"
ElseIf Not TypeOf anchor Is AecAnchorEntToGridAssembly Then
    MsgBox "Object is anchored, but not to a grid assembly.", vbExclamation
Else
    ' AdjustSizing must be set to True in order for offset change to take effect.
    If anchor.AdjustSizing = False Then
        anchor.AdjustSizing = True
        MsgBox "AdjustSizing reset from False to True"
    End If
    MsgBox "Top offset of object was: " & anchor.TopOffset, vbInformation,

    ' Prompt user to specify amount to adjust offset by.
    offset_adjust = ThisDrawing.Utility.GetReal("Enter the number of inches

    ' Change offset by specified amount.
    anchor.TopOffset = anchor.TopOffset + offset_adjust
    ThisDrawing.Regen (acActiveViewport)
    MsgBox "New top offset is: " & anchor.TopOffset, vbInformation, "TopOffset Example"
End If
Else
    MsgBox "Object selected is not an AEC entity.", vbInformation, "AdjustSizing Example"
End If

End Sub
Sub Example_AllowVariation()

    'This example will add anchor a new mass element to cell in a grid assembly.

    Dim grid As AecGridAssembly
    Dim poly As AecPolygon
    Dim pt As Variant
    Dim obj As AcadObject

    ThisDrawing.Utility.GetEntity obj, pt, "Select grid to attach to"
    If TypeOf obj Is AecGridAssembly Then
        Set grid = obj
        Set poly = ThisDrawing.ModelSpace.AddCustomObject("AecPolygon")
        Dim anchor As New AecAnchorEntToGridAssembly
        anchor.Reference = grid
        ' anchor the mass element to the first cell in the grid
        anchor.Cell = 2
        anchor.AllowVariation = True
        anchor.BottomOffset = 6
        anchor.LeftOffset = 6
        anchor.RightOffset = 6
        anchor.TopOffset = 6
        anchor.YAlignment = aecInfillAlignFrontFlush
        anchor.AdjustSizing = True
        poly.AttachAnchor anchor
    Else
        MsgBox "No Layout Grid selected", vbInformation, "Node Example"
    End If

End Sub
Sub Example AlternateName Add Aec()

' This example list the multi-view block styles in the current
drawing, and allows the user to add an alternate name to each
style (if an alternate name does not already exist).

' Use this example with a drawing that contains one or more
' multi-view block styles.

Dim app As New AecBaseApplication
Dim doc As AecBaseDocument
Dim mvblockStyles As AecMVBlockStyles
Dim mvblockStyle As AecMVBlockStyle

Dim msg As String
Dim altname As String
Dim reply As Integer

app.Init ThisDrawing.Application
Set doc = app.ActiveDocument

' Get the collection of multi-view block styles in the drawing.
Set mvblockStyles = doc.mvblockStyles

' If there are no multi-view block styles defined in the
drawing, alert user and then exit.
If mvblockStyles.count = 0 Then
    msg = "There are no multi-view block styles in the drawing."
    MsgBox msg, vbExclamation, "AlternateName Example"
    Exit Sub
End If

' List the name and alternate name of each multi-view block
'style in the current drawing. If AlternateName is blank,
'allow user to set it.
For Each mvblockStyle In mvblockStyles
msg = "MV block style name: " & mvblockStyle.Name & vbCrLf
altname = mvblockStyle.AlternateName
If altname = "" Then
  msg = msg & " There is no alternate name for " & mvblockStyle.Name & vbCrLf & "Add one at Command prompt?"
  reply = MsgBox(msg, vbYesNo, "Set alternate name at Command prompt?"

' Prompt user to enter an alternate name, if they chose
to do so.
If reply = vbYes Then
  doc.Utility.Prompt list
  altname = doc.Utility.GetString(True, "Alternate name --> ")
  mvblockStyle.AlternateName = altname
End If
Else
  msg = msg & " Alternate name: " & mvblockStyle.AlternateName & vbCrLf
  MsgBox msg, vbInformation, "AlternameName Example"
End If
Next

End Sub
Angle1 Example

Sub Example_Angle1()

' This example shows the deviation of cut plane at starting edge where zero is ' perpendicular to the building elevation line

    Dim object As Object
    Dim clip As AecClipVol
    Dim count As Integer

    ' initialize
    count = 0

    For Each object In ThisDrawing.ModelSpace

        If TypeOf object Is AecClipVol Then
            count = count + 1
            Set clip = object
            MsgBox "ClipVol " & count & " Angle1 is: " & clip.Angle1, vbInformation
        End If

    Next

    If count = 0 Then
        MsgBox "No ClipVol Present in Drawing", vbInformation, "Angle1 Example"
    End If

End Sub
Angle2 Example

Sub Example_Angle2()

'This example shows the deviation of cut plane at starting edge where zero is 'perpendicular to the building elevation line

Dim object As Object
Dim clip As AecClipVol
Dim count As Integer

'Initialize
count = 0

For Each object In ThisDrawing.ModelSpace

    If TypeOf object Is AecClipVol Then
        count = count + 1
        Set clip = object
        MsgBox "ClipVol " & count & " Angle2 is: " & clip.Angle2, vbInformation
    End If

Next

If count = 0 Then
    MsgBox "No ClipVol Present in Drawing", vbInformation, "Angle2 Example"
End If

End Sub
AngleFromNode Example

Sub Example_AngleFromNode()

    'This example will find the angle from the node for a bubble attached to a ' column grid

    Dim obj As AcadObject
    Dim bubble As AecMVBlocRef
    Dim anchor As AecAnchor
    Dim leaderAnchor As AecAnchorLeadEntToNode
    Dim pt As Variant

    ThisDrawing.Utility.GetEntity obj, pt, "Select bubble"
    If TypeOf obj Is AecMVBlocRef Then
        Set bubble = obj
        Set anchor = obj.GetAnchor
        If TypeOf anchor Is AecAnchorLeadEntToNode Then
            Set leaderAnchor = anchor
            MsgBox "Angle from node = " & ThisDrawing.Utility.AngleToString(leaderAnchor.AngleFromNode, acDegrees, 4), vbInformation, "Example AngleFromNode"
        Else
            MsgBox "Not anchored to column grid", vbInformation, "Example AngleFromNode"
        End If
    Else
        MsgBox "Not a bubble", vbInformation, "Example AngleFromNode"
    End If

End Sub
AngularAzimuth Example

Sub Example_AngularAzimuth()

' This example returns the AngularAzimuth setting for the current drawing.
Dim dbPref As AecArchBaseDatabasePreferences
Set dbPref = AecArchBaseApplication.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, "

End Sub
Sub Example_AngularDisplayFormat()

    ' This example returns the AngularDisplayFormat setting for the current drawing.

    Dim dbPref As AecArchBaseDatabasePreferences
    Set dbPref = AecArchBaseApplication.ActiveDocument.Preferences

    ' Convert the constant to a string.
    Dim strUnits As String
    Select Case dbPref.AngularDisplayFormat
        Case 0
            strUnits = "Degrees."
        Case 1
            strUnits = "Degrees/Minutes/Seconds"
        Case 2
            strUnits = "Grads."
        Case 3
            strUnits = "Radians"
        Case 4
            strUnits = "Surveyor"
    End Select

    MsgBox "The current value for AngularDisplayFormat is " & strUnits, _
        vbInformation, "AngularDisplayFormat Example"

End Sub
AngularPrecision Example

Sub Example_AngularPrecision()

' This example returns the AngularPrecision setting for the current drawing.

Dim dbPref As AecArchBaseDatabasePreferences

Set dbPref = AecArchBaseApplication.ActiveDocument.Preferences

MsgBox "The current value for AngularPrecision is " & dbPref.AngularPrecision, vbInformation, "AngularPrecision Example"

End Sub
Sub Example_Application()
    ' This example creates a line and then uses the
    ' Application property of the line to return the
    ' application name.
    Dim lineObj As AcadLine
    Dim startPoint(0 To 2) As Double
    Dim endPoint(0 To 2) As Double
    Dim myApp As AcadApplication

    ' Create a new line reference
    startPoint(0) = 0: startPoint(1) = 0: startPoint(2) = 0
    endPoint(0) = 2: endPoint(1) = 2: endPoint(2) = 0
    Set lineObj = ThisDrawing.ModelSpace.AddLine(startPoint, endPoint)
    lineObj.Update

    ' Return the application for the object
    Set myApp = lineObj.Application

    ' Display the name of the application
    MsgBox "The application name is: " & myApp.name, vbInformation, "Application Example"
End Sub
Sub Example_ApplyResize()

' This example will take a mass element anchored to a 2D Layout Grid, and
' apply a Resize Offset to it.

Dim obj As AcadObject
Dim anchor As AecAnchorEntToLayoutCell
Dim pt As Variant
ThisDrawing.Utility.GetEntity obj, pt, "Select anchored element"
If TypeOf obj Is AecMassElement Then
    Dim mass As AecMassElement
    Set mass = obj
    If TypeOf mass.GetAnchor Is AecAnchorEntToLayoutCell Then
        Set anchor = mass.GetAnchor
        anchor.ResizeOffset = -22
        anchor.ApplyResize = False
    End If
End If
End Sub
Area Example

Examples:

1 AecProfile

1 AecSlice

Sub Example_Area_AecProfile()

'This example shows the area of the profile of an AecPolygon

    Dim obj As Object
    Dim pt As Variant
    Dim poly As AecPolygon

    ThisDrawing.Utility.GetEntity obj, pt, "Select an AEC Polygon"

    If TypeOf obj Is AecPolygon Then
        Set poly = obj
        MsgBox "Profile Area: " & poly.Profile.Area, vbInformation, "Area Example"
    Else
        MsgBox "Not a Polygon or no Profile Found", vbInformation, "Area Example"
    End If

End Sub

Sub Example_Area_AecSlice()

'This example will display the area of one slice in the current drawing

    Dim slice As AecSlice
    Dim obj As Object
    Dim counter As Integer
counter = 0

For Each obj In ThisDrawing.ModelSpace
    If TypeOf obj Is AecSlice And counter < 1 Then
        counter = counter + 1
        Set slice = obj
        MsgBox "Slice Area: " & slice.Area, vbInformation, "Area Example"
    End If
Next

If counter = 0 Then
    MsgBox "No Slice in Drawing", vbInformation, "Area Example"
End If

End Sub
Sub Example_AreaDisplayUnit()

' This example returns the AreaDisplayUnit setting for the current drawing.
Dim dbPref As AecArchBaseDatabasePreferences
Set dbPref = AecArchBaseApplication.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 aecUnitSquareCentimeter
unit = "square centimeters."

Case aecUnitSquareDecimeter
    unit = "square decimeters."

Case aecUnitSquareMeters
    unit = "square meters."

End Select

MsgBox "The current value for AreaDisplayUnit is " & unit, vbInformation, "AreaDisplayUnit Example"

End Sub
Sub Example_AreaPrecision()

' This example displays the AreaPrecision setting for the current drawing.

Dim dbPref As AecArchBaseDatabasePreferences

Set dbPref = AecArchBaseApplication.ActiveDocument.preferences

MsgBox "The current value for AreaPrecision is: " & dbPref.AreaPrecision, _
        vbInformation, "AreaPrecision Example"

End Sub
Sub Example_AreaSuffix()

' This example displays the AreaSuffix setting for the current drawing.

Dim dbPref As AecArchBaseDatabasePreferences

Set dbPref = AecArchBaseApplication.ActiveDocument.preferences

MsgBox "The current value for AreaSuffix is: " & dbPref.AreaSuffix, vbInformation, "AreaSuffix Example"

End Sub
Sub Example_ArrayPolar()
    ' This example creates a circle and then performs a polar array
    ' on that circle.
    
    ' Create the circle
    Dim circleObj As AcadCircle
    Dim center(0 To 2) As Double
    Dim radius As Double
    center(0) = 2#: center(1) = 2#: center(2) = 0#
    radius = 1
    Set circleObj = ThisDrawing.ModelSpace.AddCircle(center, radius)
    ZoomAll
    MsgBox "Perform the polar array on the circle.", , "ArrayPolar Example"
    
    ' Define the polar array
    Dim noOfObjects As Integer
    Dim angleToFill As Double
    Dim basePnt(0 To 2) As Double
    noOfObjects = 4
    angleToFill = 3.14 ' 180 degrees
    basePnt(0) = 4#: basePnt(1) = 4#: basePnt(2) = 0#
    
    ' The following example will create 4 copies of an object
    ' by rotating and copying it about the point (3,3,0).
    Dim retObj As Variant
    retObj = circleObj.ArrayPolar(noOfObjects, angleToFill, basePnt)
    ZoomAll
    MsgBox "Polar array completed.", , "ArrayPolar Example"
End Sub
ArrayRectangular Example

Sub Example_ArrayRectangular()
    ' This example creates a circle and then performs
    ' a rectangular array on that circle.

    ' Create the circle
    Dim circleObj As AcadCircle
    Dim center(0 To 2) As Double
    Dim radius As Double
    center(0) = 2#: center(1) = 2#: center(2) = 0#
    radius = 0.5
    Set circleObj = ThisDrawing.ModelSpace.AddCircle(center, radius)
    ThisDrawing.Application.ZoomAll
    MsgBox "Perform the rectangular array on the circle.", , "ArrayRectangular Example"

    ' Define the rectangular array
    Dim numberOfRows As Long
    Dim numberOfColumns As Long
    Dim numberOfLevels As Long
    Dim distanceBwtnRows As Double
    Dim distanceBwtnColumns As Double
    Dim distanceBwtnLevels As Double
    numberOfRows = 5
    numberOfColumns = 5
    numberOfLevels = 2
    distanceBwtnRows = 1
    distanceBwtnColumns = 1
    distanceBwtnLevels = 1

    ' Create the array of objects
    Dim retObj As Variant
    retObj = circleObj.ArrayRectangular(numberOfRows, numberOfColumns, numberOfLevels, distanceBwtnRows, distanceBwtnColumns, distanceBwtnLevels)

    ZoomAll
    MsgBox "Rectangular array completed.", , "ArrayRectangular Example"

End Sub
Sub Example_AttachAnchor()

    ' This example will make a mass element and anchor it to a line.

    Dim mass As AecMassElement
    Dim line As AcadLine
    Dim anchor As New AecAnchorEntToCurve
    Dim pt1(0 To 2) As Double
    Dim pt2(0 To 2) As Double

    pt2(0) = 1000: pt2(1) = 1000

    ' this makes a new mass element 120 x 120 x 120 at 0,0,0
    Set mass = ThisDrawing.ModelSpace.AddCustomObject("AecMassElement")
    ' draws a line from 0,0 to 1000,1000
    Set line = ThisDrawing.ModelSpace.AddLine(pt1, pt2)

    anchor.Reference = line ' set the line as the object (curve) to anchor to
    mass.AttachAnchor anchor ' attach the anchor to the mass element

End Sub
Sub Example_AttachEntity()

' This example will make a mass element and anchor it to a line.

Dim mass As AecMassElement
Dim line As AcadLine
Dim anchor As New AecAnchorEntToCurve
Dim pt1(0 To 2) As Double
Dim pt2(0 To 2) As Double

pt2(0) = 1000: pt2(1) = 1000

' this makes a new mass element 120 x 120 x 120 at 0,0,0
Set mass = ThisDrawing.ModelSpace.AddCustomObject("AecMassElement")
' draws a line from 0,0 to 1000,1000
Set line = ThisDrawing.ModelSpace.AddLine(pt1, pt2)

anchor.Reference = line ' set the line as the object (curve) to anchor to
anchor.AttachEntity mass ' attach the mass element to the anchor

End Sub
Sub Example_BasePoint()

' This example displays the BasePoint setting for the current drawing.

Dim dbPref As AecArchBaseDatabasePreferences
Set dbPref = AecArchBaseApplication.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 AecArchBaseDatabasePreferences
Set dbPref = AecArchBaseApplication.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
Sub Example_Blocks()
    ' This example first finds the blocks collection
    ' using the Blocks property. It then adds a new
    ' block to the block collection.

    Dim blkColl As AcadBlocks
    Dim newBlock As AcadBlock
    Dim insertionPnt(0 To 2) As Double

    ' Get the Blocks collection
    Set blkColl = ThisDrawing.Blocks

    ' Define the insertion point for the new block
    insertionPnt(0) = 0#: insertionPnt(1) = 0#: insertionPnt(2) = 0#

    ' Add the new block into the blocks collection
    Set newBlock = blkColl.Add(insertionPnt, "TEST")
    MsgBox "A block called " & newBlock.name & " has been added to the blocks collection", vbInformation, "Blocks Example"
End Sub
Sub Example_BottomOffset()

' This example modifies the bottom offset of the selected object
' in relation to its anchor point on the grid assembly. It
' prompts the user to select the object to be modified,
' and then it prompts the user to specify the number of inches
' to adjust the offset by. The number can be either positive
' or negative.

' Use this example with a drawing that contains a window
' assembly and one or more AEC objects attached to the
' assembly.

Dim ent As AcadEntity
Dim geo As AecGeo
Dim anchor As AecAnchor
Dim offset As String
Dim offset_adjust As Double

On Error Resume Next        ' Handle errors in code.

' Prompt user to select an object.
ThisDrawing.Utility.GetEntity ent, pt, "Select object anchored to window asse

' Make sure user selected an AEC object, and that the object
' is anchored to a grid assembly.
If ent Is Nothing Then
    MsgBox "Nothing was selected.", vbExclamation, "BottomOffset Example"
ElseIf TypeOf ent Is AecGeo Then
    Set geo = ent

    ' Get the anchor the object is attached to.
    Set anchor = geo.GetAnchor
    On Error GoTo 0
    If anchor Is Nothing Then

    ' ThisDrawing.Utility.GetEntity ent, pt, "Select object anchored to window asse
MsgBox "Selected object is not anchored.", vbExclamation, "BottomOffset Example"

ElseIf Not TypeOf anchor Is AecAnchorEntToGridAssembly Then
    MsgBox "Object is anchored, but not to a grid assembly.", vbExclamation
Else
    ' AdjustSizing must be set to True in order for offset change to take effect.
    anchor.AdjustSizing = True
    MsgBox "Bottom offset of object was: " & anchor.BottomOffset, vbInformation, "BottomOffset Example"

    ' Prompt user to specify amount to adjust offset by.
    offset_adjust = ThisDrawing.Utility.GetReal("Enter the number of inches to adjust offset by:")
    anchor.AdjustSizing = True
    anchor.BottomOffset = anchor.BottomOffset + offset_adjust
    ThisDrawing.Regen (acActiveViewport)
    MsgBox "New bottom offset is: " & anchor.BottomOffset, vbInformation, "BottomOffset Example"
End If
Else
    MsgBox "Object selected is not an AEC entity.", vbInformation, "BottomOffset Example"
End If

End Sub
Sub Example_Bounds()

' This example will display the bounds of the first profile in the drawing.

Dim doc As AecArchBaseDocument
Dim cProfileStyles As AecProfileStyles
Dim profileStyle As AecProfileStyle
Dim profile As AecProfile

Set doc = AecArchBaseApplication.ActiveDocument
Set cProfileStyles = doc.ProfileStyles
If cProfileStyles.Count > 0 Then
    Set profileStyle = cProfileStyles.Item(0)
    Set profile = profileStyle.profile
    MsgBox "Profile Bounds: " & vbCrLf & 
            "Top Left: " & Format(profile.bounds(0), "0.000") & ", " & Format(profile.bounds(1), "0.000"), ", " & Format(profile.bounds(2), "0.000") & ", " & Format(profile.bounds(3), "0.000"), vbInformation, "Bounds Example"
Else
    MsgBox "No AEC Profiles in the drawing", vbInformation, "Bounds Example"
End If

End Sub
Sub Example_Cell()

'This example will add anchor a new mass element to cell in a 2D layout grid.

Dim grid As AecGridAssembly
Dim mass As AecMassElement
Dim pt As Variant
Dim obj As AcadObject

ThisDrawing.Utility.GetEntity obj, pt, "Select grid to attach to"
If TypeOf obj Is AecGridAssembly Then
    Set grid = obj
    Set mass = ThisDrawing.ModelSpace.AddCustomObject("AecMassElement"
    Dim anchor As New AecAnchorEntToGridAssembly
    anchor.Reference = grid
    ' anchor the mass element to the first cell in the grid
    anchor.Cell = 1
    mass.AttachAnchor anchor
Else
    MsgBox "No Layout Grid selected", vbInformation, "Node Example"
End If

End Sub
Sub Example_CenterOnNode()

    ' This example attaches a Mass Element to a 2D Layout Grid, and centers the
    ' the node.

    Dim obj As AcadObject
    Dim pt As Variant
    ThisDrawing.Utility.GetEntity obj, pt, "Select Layout Grid"

    If TypeOf obj Is AecLayoutGrid2D Then
        Dim grid As AecLayoutGrid2D
        Set grid = obj
        Dim mass As AecMassElement
        Set mass = ThisDrawing.ModelSpace.AddCustomObject("AecMassElement"
        Dim anchor As New AecAnchorEntToLayoutNode
        anchor.Reference = grid
        anchor.Node = 1
        anchor.CenterOnNode = True
        mass.AttachAnchor anchor
    Else
        MsgBox "No Layout Grid selected", vbInformation, "CenterOnNode Example"
    End If

End Sub
Sub Example_Centroid()

' This example will display the centroid of the first profile in the drawing.

Dim doc As AecArchBaseDocument
Dim cProfileStyles As AecProfileStyles
Dim profileStyle As AecProfileStyle
Dim profile As AecProfile

Set doc = AecArchBaseApplication.ActiveDocument
Set cProfileStyles = doc.ProfileStyles
If cProfileStyles.Count > 0 Then
    Set profileStyle = cProfileStyles.Item(0)
    Set profile = profileStyle.profile
    MsgBox "Profile Centroid: " & profile.Centroid(0) & ", " & profile.Centroid(1), vbInformation, "Centroid Example"
Else
    MsgBox "No AEC Profiles in the drawing", vbInformation, "Centroid Example"
End If

End Sub
Sub Example_Clear_AecProfile()

' This example copies a profile and uses the Clear method to
' remove all the rings from the new copy.

' The example looks for a profile style named
' "Hinged - Double - Full Lite." If you do not have this style
' in the current drawing, change the name of the style to one
' that exists in your drawing.

Dim doc As AecBaseDocument
Dim app As New AecBaseApplication
Dim profileStyle As AecProfileStyle
Dim profile As New AecProfile
Dim copied_profile As New AecProfile

Dim profileName As String
Dim msg As String

app.Init ThisDrawing.Application
Set doc = app.ActiveDocument

' Get the specified profile style.
On Error Resume Next
profileName = "Hinged - Double - Full Lite"
Set profileStyle = doc.ProfileStyles.Item(profileName)
On Error Resume Previous

' If an error occurred, the profile probably doesn't exist.
If Err.Number  0 Then
    MsgBox "Profile " & profileName & " does not exist.", vbExclamation, "Clear Example"
    Exit Sub
End If

' Set an AecProfile object to the style you retrieved.
Set profile = profileStyle.profile
' Copy the profile.
copied_profile.CopyFrom profile
msg = "Copied profile had " & copied_profile.Rings.count & "rings." & vbCrLf

' Remove the rings in the copied profile using the Clear method.
copied_profile.Clear
msg = msg & "After Clear, the copied profile has " & copied_profile.Rings.cou

MsgBox msg, vbInformation, "Clear Example"

End Sub

End Sub
Sub Example_Contains()

' This example will check if an AecRing contains another AecRing.

On Error Resume Next
Dim pointList1(0 To 9) As Double
Dim pointList2(0 To 9) As Double
pointList1(0) = 0: pointList1(1) = 0
pointList1(2) = 1: pointList1(3) = 0
pointList1(4) = 1: pointList1(5) = 1
pointList1(6) = 0: pointList1(7) = 1
pointList1(8) = 0: pointList1(9) = 0
pointList1(0) = 0.25: pointList1(1) = 0.25
pointList1(2) = 0.75: pointList1(3) = 0.25
pointList1(4) = 0.75: pointList1(5) = 0.75
pointList1(6) = 0.25: pointList1(7) = 0.75
pointList1(8) = 0.25: pointList1(9) = 0.25

Dim ring1 As AecRing
Dim ring2 As AecRing
Dim profile1 As New AecProfile

Dim doc As AecArchBaseDocument
Set doc = AecArchBaseApplication.ActiveDocument
Dim cprofiles As AecProfileStyles
Dim profileStyle As AecProfileStyle
Dim profile As AecProfile

Set cprofiles = doc.ProfileStyles
Set profileStyle = cprofiles.Item("ContainsRing")
If profileStyle Is Nothing Then
    Set profileStyle = cprofiles.Add("ContainsRing")
End If
Set ring1 = profile1.Rings.Add
Set ring2 = profile1.Rings.Add
ring1.FromPoints (pointList1)
ring2.FromPoints (pointList2)
MsgBox "Ring1 contains Ring2 is: " & ring1.Contains(ring2)

End Sub
Sub Example_ContourLinesPerSurface()
    ' This example returns the current setting of
    ' ContourLinesPerSurface. It then changes the value, and finally
    ' it resets the value back to the original setting.

    Dim currContourLinesPerSurface As Integer
    Dim newContourLinesPerSurface As Integer

    ' Retrieve the current ContourLinesPerSurface value
    currContourLinesPerSurface = ThisDrawing.preferences.ContourLinesPerSurf:
    MsgBox "The current value for ContourLinesPerSurface is " & currContourLi

    ' Change the value for ContourLinesPerSurface
    newContourLinesPerSurface = 2001
    ThisDrawing.preferences.ContourLinesPerSurface = newContourLinesPerSurf
    MsgBox "The new value for ContourLinesPerSurface is " & newContourLines

    ' Reset ContourLinesPerSurface to its original value
    ThisDrawing.preferences.ContourLinesPerSurface = currContourLinesPerSurf:
    MsgBox "The ContourLinesPerSurface value is reset to " & currContourLines
End Sub
ConvertToCurrentAreaDisplay Example

Sub Example_ConvertToCurrentAreaDisplay()

' This example returns the value of ConvertToCurrentAreaDisplay
' for the current drawing.

Dim dbPref As AecArchBaseDatabasePreferences
Set dbPref = AecArchBaseApplication.ActiveDocument.preferences

' Set the area display units to square yards
dbPref.AreaDisplayUnit = aecUnitSquareYard

Dim SourceArea As Double
Dim TargetArea As Double

SourceArea = 36.0 * 36.0 'Square units, assumes drawing set to inch units
TargetArea = dbPref.ConvertToCurrentAreaDisplay(SourceArea)

MsgBox "The area to convert is " & sourcearea & " square inches." & vbCrLf & 
"The AreaDisplayUnit is set to square yard." & vbCrLf & _
"The value for ConvertToCurrentAreaDisplay is " & TargetArea & " square
vbInformation, "ConvertToCurrentAreaDisplay Example"

End Sub
ConvertToCurrentVolumeDisplay Example

Sub Example_ConvertToCurrentVolumeDisplay()

' This example returns the value of ConvertToCurrentVolumeDisplay
' for the current drawing.

Dim dbPref As AecArchBaseDatabasePreferences
Set dbPref = AecArchBaseApplication.ActiveDocument.preferences

' Set the volume display units to cubic yards
dbPref.VolumeDisplayUnit = aecUnitCubicYard

Dim SourceArea As Double
Dim TargetArea As Double

SourceArea = 36# * 36# * 36#         'Cubic inches
TargetArea = dbPref.ConvertToCurrentVolumeDisplay(SourceArea)

MsgBox "The area to convert is " & SourceArea & " cubic inches." & vbCrLf & "The VolumeDisplayUnit is set to cubic yard." & vbCrLf & "The value for ConvertToCurrentVolumeDisplay is " & TargetArea & " " & vbCrLf
vbInformation, "ConvertToCurrentVolumeDisplay Example"

End Sub
Sub Example_CoordinatePrecision()

' This example returns the CoordinatePrecision setting for the current drawing.

Dim dbPref As AecArchBaseDatabasePreferences

Set dbPref = AecArchBaseApplication.ActiveDocument.Preferences

MsgBox "The current value for CoordinatePrecision is: " & dbPref.CoordinatePrecision, vbInformation, "CoordinatePrecision Example"

End Sub
Copy Example

Sub Example_Copy()
  ' This example creates a circle and then copies that circle. The new circle is then moved and colored red.
  
  ' Create the circle
  Dim circleObj As AcadCircle
  Dim center(0 To 2) As Double
  Dim radius As Double
  center(0) = 2#: center(1) = 2#: center(2) = 0#
  radius = 0.5
  Set circleObj = ThisDrawing.ModelSpace.AddCircle(center, radius)
  ZoomAll
  MsgBox "Copy the circle.", , "Copy Example"
  
  ' Copy the circle
  Dim copyCircleObj As AcadCircle
  Set copyCircleObj = circleObj.Copy()
  
  ' Define the points that make up the move vector
  Dim point1(0 To 2) As Double
  Dim point2(0 To 2) As Double
  point1(0) = 0: point1(1) = 0: point1(2) = 0
  point2(0) = 2: point2(1) = 0: point2(2) = 0
  
  MsgBox "Move the copied circle 2 units in the X direction.", , "Copy Example"
  
  ' Move the circle and color it
  copyCircleObj.Move point1, point2
  copyCircleObj.Color = acRed
  
  ZoomAll
  MsgBox "Move completed.", , "Copy Example"

End Sub
CopyFrom Example

Sub Example_CopyFrom_AecProfile()

' This example copies an existing profile using the CopyFrom method.

' The example looks for a profile style named
' "Hinged - Double - Full Lite." If you do not have this style
' in the current drawing, change the name of the style to one
' that exists in your drawing.

Dim doc As AecBaseDocument
Dim app As New AecBaseApplication
Dim profileStyle As AecProfileStyle
Dim profile As New AecProfile
Dim copied_profile As New AecProfile

Dim profileName As String
Dim msg As String

app.Init ThisDrawing.Application
Set doc = app.ActiveDocument

' Get the specified profile style.
On Error Resume Next
profileName = "Hinged - Double - Full Lite"
Set profileStyle = doc.ProfileStyles.Item(profileName)

' If an error occurred, the profile probably doesn't exist.
If Err.Number ≠ 0 Then
    MsgBox "Profile " & profileName & " does not exist.", vbExclamation, "CopyFrom Example"
    Exit Sub
End If

' Set an AecProfile object to the style you retrieved.
Set profile = profileStyle.profile

' Copy the profile.
copied_profile.CopyFrom profile

msg = "Copied profile had " & copied_profile.Rings.count & " rings." & vbCrLf
MsgBox msg, vbInformation, "CopyFrom Example"

End Sub
CopyObjects Example

Sub Example_CopyObjects()
    ' This example creates a Circle object and uses the CopyObjects method to make a copy of the new Circle.

    Dim DOC1 As AcadDocument
    Dim circleObj1 As AcadCircle, circleObj2 As AcadCircle
    Dim circleObj1Copy As AcadCircle, circleObj2Copy As AcadCircle
    Dim centerPoint(0 To 2) As Double
    Dim radius1 As Double, radius2 As Double
    Dim radius1Copy As Double, radius2Copy As Double
    Dim objCollection(0 To 1) As Object
    Dim retObjects As Variant

    ' Define the Circle object
    centerPoint(0) = 0: centerPoint(1) = 0: centerPoint(2) = 0
    radius1 = 5#: radius2 = 7#
    radius1Copy = 1#: radius2Copy = 2#

    ' Create a new drawing
    Set DOC1 = Documents.Add

    ' Add two circles to the drawing
    Set circleObj1 = DOC1.ModelSpace.AddCircle(centerPoint, radius1)
    Set circleObj2 = DOC1.ModelSpace.AddCircle(centerPoint, radius2)
    ThisDrawing.Application.ZoomAll

    ' Copy objects
    ',

    ' First put the objects to be copied into a form compatible with CopyObjects
    Set objCollection(0) = circleObj1
    Set objCollection(1) = circleObj2

    ' Copy object and get back a collection of the new objects (copies)
    retObjects = DOC1.CopyObjects(objCollection)

    ' Get newly created object and apply new properties to the copies
Set circleObj1Copy = retObjects(0)
Set circleObj2Copy = retObjects(1)

circleObj1Copy.radius = radius1Copy
circleObj1Copy.Color = acRed
circleObj2Copy.radius = radius2Copy
circleObj2Copy.Color = acRed

ThisDrawing.Application.ZoomAll

MsgBox "Circles copied."
End Sub
Count Example

Sub Example_Count()

    ' This example will display the number of MultiView Block Styles in the current drawing.

    Dim doc As AecArchBaseDocument
    Dim cMvBlockStyles As AecMVBlocKStyles

    Set doc = AecArchBaseApplication.ActiveDocument
    Set cMvBlockStyles = doc.MVBlocKStyles
    MsgBox "Number of MVBlocK Styles in this drawing is: " & cMvBlockStyles.

End Sub
Sub Example_Database()
   ' This example references the Database object obtained from the model space.
   ' We then display information from the Database object.

   Dim Database As AcadDatabase

   ' Attach to Database object
   Set Database = ThisDrawing.ModelSpace.Database
   MsgBox "We now have access to the properties and methods of the Database object!"

   ' Retrieve the number of Blocks in this database
   MsgBox "The number of Blocks in this database is: " & Database.Blocks.count

   ' Release object
   Set Database = Nothing
   MsgBox "The Database object has been released!"
End Sub
Sub Example_DatabaseScale()

' This example returns the Database Scale setting for the current drawing.

Dim dbPref As AecArchBaseDatabasePreferences

Set dbPref = AecArchBaseApplication.ActiveDocument.Preferences

MsgBox "The current value for DatabaseScale is: " & dbPref.DatabaseScale, _
   vbInformation, "DatabaseScale Example"

End Sub
Sub Example_Delete()

    ' This example prompts users to select a drawing object, then
    ' deletes that object. It then issues the Undo command to
    ' reverse the delete.

    Dim obj As AcadObject
    Dim ent As AecEntity
    Dim pickPt As Variant

    On Error Resume Next
    ThisDrawing.Utility.GetEntity obj, pickPt, "Please select an AEC object:" & vbCrLf
    On Error GoTo 0

    ' Verify that an AEC object was selected.
    If obj Is Nothing Then
        MsgBox "Nothing selected.", vbExclamation, "Delete Example"
        Exit Sub
    ElseIf Not (TypeOf obj Is AecEntity) Then
        MsgBox "The object you selected is not an AEC object.", vbExclamation, "Delete Example"
        Exit Sub
    Else
        Set ent = obj
        Set obj = Nothing
    End If

    ' Delete the selected object.
    ent.Delete
    ThisDrawing.Regen (acActiveViewport)
    MsgBox "The selected object was temporarily deleted.", vbExclamation, "Delete Example"

    ' Undo the delete.
    ThisDrawing.SendCommand "_undo" & vbCrLf & vbCrLf

End Sub
Depth Example

Examples:

- AecLayoutGrid2D
- AecLayoutGrid3D
- AecMassElement

Sub Example_Depth_AecLayoutGrid2D()

  'This example displays the depth of a 2D layout grid

  Dim obj As Object
  Dim pt As Variant
  Dim grid As AecLayoutGrid2D
  Dim msg As String
  Dim angAsString As String

  ' Ask user to select a grid.
  ThisDrawing.Utility.GetEntity obj, pt, "Select a 2D Layout Grid"

  If TypeOf obj Is AecLayoutGrid2D Then
    Set grid = obj
    Set obj = Nothing

    ' If the grid is radial, then Depth is an angle, in radians.
    ' There is no way to check the grid type, because there is
    ' no Shape property exposed. So this example displays both
    ' the raw value (to three decimal places) and the value in
    ' degrees (converted from radians).
    msg = msg & "Depth is " & Format(grid.Depth, "0.000")

    angAsString = ThisDrawing.Utility.AngleToString(grid.Depth, acDegrees, 2)
    msg = msg & " (if radial, this is " & angAsString & " degrees)" & vbCrLf

  ' Convert the radian value to degrees with a precision of 2.
  angAsString = ThisDrawing.Utility.AngleToString(grid.Depth, acDegrees, 2
  msg = msg & " (if radial, this is " & angAsString & " degrees)" & vbCrLf
Sub Example_Depth_AecLayoutGrid3D()

' This example displays the depth of a 3D layout grid
Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid3D
ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid" If TypeOf obj Is AecLayoutGrid3D Then Set grid = obj MsgBox "Grid Depth is: " & grid.Depth, vbInformation, "Depth Example" Else MsgBox "Not a 3D Layout Grid", vbExclamation, "Depth Example" End If

End Sub

Sub Example_Depth_AecMassElement()

' This example shows the size of the mass element in its relative Y direction

Dim object As Object
Dim mass As AecMassElement
Dim count As Integer

' Initialize
count = 0
For Each object In ThisDrawing.ModelSpace
    If TypeOf object Is AecMassElement Then
        count = count + 1
        Set mass = object
        MsgBox "Mass Element " & count & " Depth is: " & mass.Depth, vbInformation, "Depth Example"
    End If
End If
Next

If count = 0 Then
    MsgBox "No Mass Elements Present in Drawing", vbInformation, "Depth E
End If

End Sub
Sub Example_Description()

' This example will display the description of an AEC Entity.

Dim obj As AcadObject
Dim pt As Variant
ThisDrawing.Utility.GetEntity obj, pt, "Select AEC Object"
If TypeOf obj Is AecGeo Then
    Dim AecGeo As AecGeo
    ' AecGeo is the base class for all the drawn AEC Entities
    Set AecGeo = obj
    MsgBox "AEC Description:" & AecGeo.Description, vbInformation, "Description Example"
Else
    MsgBox "No AEC Entity selected", vbInformation, "Description Example"
End If

End Sub
Sub Example_Dictionaries()
    ' This example finds the current dictionary collection and
    ' adds a new dictionary to that collection.

    Dim dict As AcadDictionary
    Set dict = ThisDrawing.Dictionaries.Add("TEST")
    MsgBox "A new dictionary called ", dict.name, " has been added to the dic:"
End Sub
Sub Example_DimStyles()
    ' This example adds a new dimension style.
    Dim dimStyle As AcadDimStyle
    Set dimStyle = ThisDrawing.DimStyles.Add("TEST")
    MsgBox "A new dimension style called " & dimStyle.name & " has been added.
End Sub
Sub Example_DisplaySilhouette()
    ' This example returns the current setting of
    ' DisplaySilhouette. It then changes the value, and finally
    ' it resets the value back to the original setting.

    Dim currDisplaySilhouette As Boolean

    ' Retrieve the current DisplaySilhouette value
    currDisplaySilhouette = ThisDrawing.preferences.DisplaySilhouette
    MsgBox "The current value for DisplaySilhouette is " & preferences.DisplaySi

    ' Change the value for DisplaySilhouette
    ThisDrawing.preferences.DisplaySilhouette = Not (currDisplaySilhouette)
    MsgBox "The new value for DisplaySilhouette is " & preferences.DisplaySi

    ' Reset DisplaySilhouette to its original value
    ThisDrawing.preferences.DisplaySilhouette = currDisplaySilhouette
    MsgBox "The DisplaySilhouette value is reset to " & preferences.DisplaySi
End Sub
Sub Example_Document()
    ' This example creates a circle in model space and then
    ' finds the name of the document that the circle resides in.
    Dim circleObj As AcadCircle
    Dim centerPoint(0 To 2) As Double
    Dim radius As Double

    ' Define the circle
    centerPoint(0) = 0#: centerPoint(1) = 0#: centerPoint(2) = 0#
    radius = 5#

    ' Create the Circle object in model space
    Set circleObj = ThisDrawing.ModelSpace.AddCircle(centerPoint, radius)
    ZoomAll

    ' Find the document name for the circle
    Dim currDoc As AcadDocument
    Set currDoc = circleObj.Document

    MsgBox "The circle resides in: " & currDoc.name
End Sub
Sub Example_EdgeWidth()

    ' This example will display the edge width of the selected AecPolygon

    Dim obj As AcadObject
    Dim pt As Variant

    ThisDrawing.Utility.GetEntity obj, pt, "Select AecPolygon"

    If TypeOf obj Is AecPolygon Then
        Dim polygon As AecPolygon
        Dim polygonStyle As AecPolygonStyle
        Set polygon = obj
        Set polygonStyle = polygon.Style
        MsgBox "Polygon EdgeWidth = " & polygonStyle.EdgeWidth, vbInformation, "EdgeWidth Example"
    Else
        MsgBox "No AecPolygon Selected", vbInformation, "EdgeWidth Example"
    End If

End Sub
Sub Example_EdgeWidthJustify()

    ' This example will display the edge width justification of the selected AecPolygon

    Dim obj As AcadObject
    Dim pt As Variant

    ThisDrawing.Utility.GetEntity obj, pt, "Select AecPolygon"

    If TypeOf obj Is AecPolygon Then
        Dim polygon As AecPolygon
        Dim polygonStyle As AecPolygonStyle
        Set polygon = obj
        Set polygonStyle = polygon.Style
        Select Case polygonStyle.EdgeWidthJustify
            Case aecPolygonEdgeWidthJustificationCenter
                MsgBox "Polygon EdgeWidthJustify CENTER", vbInformation, "EdgeWidthJustify Example"
            Case aecPolygonEdgeWidthJustificationIn
                MsgBox "Polygon EdgeWidthJustify is IN", vbInformation, "EdgeWidthJustify Example"
            Case aecPolygonEdgeWidthJustificationOut
                MsgBox "Polygon EdgeWidthJustify is OUT", vbInformation, "EdgeWidthJustify Example"
        End Select
    Else
        MsgBox "No AecPolygon Selected", vbInformation, "EdgeWidthJustify Example"
    End If

End Sub
ElevationModelSpace Example

Sub Example_ElevationModelSpace()
    ' This example changes the model space elevation of the current drawing
    ' and then resets it to the original value again.

    Dim currElevation As Double

    currElevation = ThisDrawing.ElevationModelSpace
    MsgBox "The current model space elevation is " & ThisDrawing.ElevationModelSpace, vbInformation, "ElevationModelSpace Example"

    ' Change the elevation
    ThisDrawing.ElevationModelSpace = currElevation + 2
    MsgBox "The new model space elevation is " & ThisDrawing.ElevationModelSpace, vbInformation, "ElevationModelSpace Example"

    ' Reset the elevation to its original value
    ThisDrawing.ElevationModelSpace = currElevation
    MsgBox "The model space elevation is reset to " & ThisDrawing.ElevationModelSpace, vbInformation, "ElevationModelSpace Example"

End Sub
Sub Example_ElevationPaperSpace()
    ' This example changes the paperspace elevation of the current drawing
    ' and then resets it to the original value again.
    Dim currElevation As Double

    currElevation = ThisDrawing.ElevationPaperSpace
    MsgBox "The current paper space elevation is " & ThisDrawing.ElevationPaperSpace, vbInformation, "ElevationPaperSpace Example"

    ' Change the elevation
    ThisDrawing.ElevationPaperSpace = currElevation + 2
    MsgBox "The new paperspace elevation is " & ThisDrawing.ElevationPaperSpace, vbInformation, "ElevationPaperSpace Example"

    ' Reset the elevation to its original value
    ThisDrawing.ElevationPaperSpace = currElevation
    MsgBox "The paperspace elevation is reset to " & ThisDrawing.ElevationPaperSpace, vbInformation, "ElevationPaperSpace Example"

End Sub
Sub Example_ElevationPrecision()

' This example returns the ElevationPrecision setting for the current drawing.

Dim dbPref As AecDatabasePreferences

Set dbPref = AecArchBaseApplication.ActiveDocument.Preferences

MsgBox "The current value for ElevationPrecision is: " & dbPref.ElevationPrecision,
       vbInformation, "ElevationPrecision Example"

End Sub
EndOffset Example

Sub Example_EndOffset()

'This example displays the end offset of a selected AEC Layout Curve Object

Dim obj As Object
Dim pt As Variant
Dim layoutCurve As AecLayoutCurve

ThisDrawing.Utility.GetEntity obj, pt, "Select a Node on an AEC Layout Curve"

If TypeOf obj Is AecLayoutCurve Then
    Set layoutCurve = obj
    MsgBox "EndOffset is: " & layoutCurve.EndOffset, vbInformation, "EndOffset Example"
Else
    MsgBox "Not a AEC Layout Curve", vbExclamation, "EndOffset Example"
End If

End Sub
Sub Example_ExpandKey()

' This example shows how to expand a layer key for the current layer standard

    Dim doc As AecArchBaseDocument
    Dim dbPref As AecArchBaseDatabasePreferences
    Dim cLayerKeyStyles As AecLayerKeyStyles
    Dim layerKeyStyle As AecLayerKeyStyle
    Dim cLayerKeys As AecLayerKeys
    Dim layerKey As AecLayerKey

    Set doc = AecArchBaseApplication.ActiveDocument
    Set cLayerKeyStyles = doc.LayerKeyStyles
    Set dbPref = doc.Preferences
    ' Sets the layer key style to the current layer standard
    Set layerKeyStyle = cLayerKeyStyles.Item(dbPref.LayerStandard)

    MsgBox "Layer Key WALL expands to: " & layerKeyStyle.ExpandKey("WALL")
    vbInformation, "Expand Example"

End Sub
ExportFreeForm Example

Sub Example_ExportFreeForm()

' This example creates a mass element, after prompting users
' for its insertion point, and then exports the mass element
' boundaries to a file.
' You can run this code in conjunction with Example_ImportFreeForm,
' which imports the saved boundaries.

Dim massElement As AecMassElement
Dim obj As AcadObject
Dim center_at_origin As Boolean

Dim object_selected As Boolean
object_selected = False

' Create a new mass element with a pyramid shape.
Set massElement = ThisDrawing.ModelSpace.AddCustomObject("AecMassEl
massElement.Type = aecMassElementTypePyramid

'Specify a location for the mass element.
pt = ThisDrawing.Utility.GetPoint(, "Select the insertion point:")
If Err.Number  0 Then
    MsgBox ("error when getting a point." & vbCrLf)
    Exit Sub
End If

massElement.Location = pt
massElement.Rotation = 0

'Set the pyramid's dimensions to 12' by 12' by 12'
massElement.Width = 144
massElement.Depth = 144
massElement.Height = 144

' Indicate that original coordinates of the boundary
' representation are to be saved.
center_at_origin = True

' Export the mass element boundaries.
massElement.ExportFreeForm "c:\temp\freeform-massElement", center_at_ori

End Sub
Sub Example_FacetDeviation()

' This example displays the FacetDeviation setting for the current drawing.

Dim dbPref As AecArchBaseDatabasePreferences

Set dbPref = AecArchBaseApplication.ActiveDocument.preferences

MsgBox "The setting for FacetDeviation is: " & dbPref.FacetDeviation, _
    vbInformation, "FacetDeviation Example"

End Sub
FlipX Example

Sub Example_FlipX()

' This example will flip an object that uses an AecAnchorEntToCurve about the

Dim obj As AcadObject
Dim pnt As Variant
ThisDrawing.Utility.GetEntity obj, pnt, vbCrLf & "Select a Geo anchored to a

If TypeOf obj Is AecGeo Then
Dim geo As AecGeo
Set geo = obj
Dim anchor As AecAnchor
Set anchor = geo.GetAnchor
If TypeOf anchor Is AecAnchorEntToCurve Then
    Dim curveAnchor As AecAnchorEntToCurve
    Set curveAnchor = anchor
    If curveAnchor.FlipX Then
        MsgBox "FlipX is True", vbInformation, "FlipX Example"
    Else
        MsgBox "FlipX is False", vbInformation, "FlipX Example"
    End If
' The following line will flip the current anchor
    curveAnchor.FlipX = Not curveAnchor.FlipX
Else
    MsgBox "Anchor not of type AecAnchorEntToCurve", vbExclamation, "I
End If
Else
    MsgBox "Not an AecGeo Object", vbExclamation, "FlipX Example"
End If

End Sub
FlipY Example

Sub Example_FlipY()

' This example will flip an object that uses an AecAnchorEntToCurve about the Y axis.

Dim obj As AcadObject
Dim pnt As Variant
ThisDrawing.Utility.GetEntity obj, pnt, vbCrLf & "Select a Geo anchored to a Curve"

If TypeOf obj Is AecGeo Then
    Dim geo As AecGeo
    Set geo = obj
    Dim anchor As AecAnchor
    Set anchor = geo.GetAnchor
    If TypeOf anchor Is AecAnchorEntToCurve Then
        Dim curveAnchor As AecAnchorEntToCurve
        Set curveAnchor = anchor
        If curveAnchor.FlipY Then
            MsgBox "FlipY is True", vbInformation, "FlipY Example"
        Else
            MsgBox "FlipY is False", vbInformation, "FlipY Example"
        End If
        ' The following line will flip the current anchor
        curveAnchor.FlipY = Not curveAnchor.FlipY
    Else
        MsgBox "Anchor not of type AecAnchorEntToCurve", vbExclamation, "FlipY Example"
    End If
Else
    MsgBox "Not an AecGeo Object", vbExclamation, "FlipY Example"
End If

End Sub
FlipZ Example

Sub Example_FlipZ()

' This example will flip an object that uses an AecAnchorEntToCurve about the Z-axis.

Dim obj As AcadObject
Dim pnt As Variant
ThisDrawing.Utility.GetEntity obj, pnt, vbCrLf & "Select a Geo anchored to a Curve"

If TypeOf obj Is AecGeo Then
    Dim geo As AecGeo
    Set geo = obj
    Dim anchor As AecAnchor
    Set anchor = geo.GetAnchor
    If TypeOf anchor Is AecAnchorEntToCurve Then
        Dim curveAnchor As AecAnchorEntToCurve
        Set curveAnchor = anchor
        If curveAnchor.FlipZ Then
            MsgBox "FlipZ is True", vbInformation, "FlipZ Example"
        Else
            MsgBox "FlipZ is False", vbInformation, "FlipZ Example"
        End If
        ' The following line will flip the current anchor
        curveAnchor.FlipZ = Not curveAnchor.FlipZ
    Else
        MsgBox "Anchor not of type AecAnchorEntToCurve", vbExclamation, "FlipZ Example"
    End If
Else
    MsgBox "Not an AecGeo Object", vbExclamation, "FlipZ Example"
End If

End Sub
Sub Example_FromPoints()

' This example will make a new profile from a set of points.

  On Error Resume Next
  Dim pointList(0 To 7) As Double
  pointList(0) = 0.5: pointList(1) = 0.5
  pointList(2) = 0.067: pointList(3) = -0.25
  pointList(4) = 0.933: pointList(5) = -0.25
  pointList(6) = 0.5: pointList(7) = 0.5

  Dim ring As AecRing
  Dim profile As New AecProfile

  Dim doc As AecArchBaseDocument
  Set doc = AecArchBaseApplication.ActiveDocument
  Dim cprofiles As AecProfileStyles
  Dim profileStyle As AecProfileStyle

  Set cprofiles = doc.ProfileStyles
  Set profileStyle = cprofiles.Item("FromPoints")
  If profileStyle Is Nothing Then
    Set profileStyle = cprofiles.Add("FromPoints")
  End If
  Set ring = profile.Rings.Add

  ring.FromPoints pointList
  Set profileStyle.profile = profile

End Sub
FromPolyline Example

Sub Example_FromPolyline()

' This example creates AEC Profile from a 2D Polyline.

Dim plineObj As AcadPolyline
Dim points(0 To 14) As Double

' Define the 2D polyline points
' The 3rd element is ignored
points(0) = 1: points(1) = 1: points(2) = 0:
points(3) = 1: points(4) = 2: points(5) = 0:
points(6) = 2: points(7) = 2: points(8) = 0:
points(9) = 3: points(10) = 2: points(11) = 0:
points(12) = 4: points(13) = 4: points(14) = 0:

' Create a 2D Polyline object in model space
Set plineObj = ThisDrawing.ModelSpace.AddPolyline(points)
On Error Resume Next

Dim ring As AecRing
Dim profile As New AecProfile
Dim doc As AecArchBaseDocument
Set doc = AecArchBaseApplication.ActiveDocument
Dim cprofiles As AecProfileStyles
Dim profileStyle As AecProfileStyle

Set cprofiles = doc.ProfileStyles
Set profileStyle = cprofiles.Item("FromPolyline")
If profileStyle Is Nothing Then
    Set profileStyle = cprofiles.Add("FromPolyline")
End If
Set ring = profile.Rings.Add

ring.FromPolyline plineObj

plineObj.delete
Set profileStyle.profile = profile

End Sub
Sub Example_GenerateLayer()

    'This example shows how to generate a layer with the current layer standard

    Dim doc As AecArchBaseDocument
    Dim dbPref As AecArchBaseDatabasePreferences
    Dim cLayerKeyStyles As AecLayerKeyStyles
    Dim layerKeyStyle As AecLayerKeyStyle
    Dim cLayerKeys As AecLayerKeys
    Dim layerKey As AecLayerKey

    Set doc = AecArchBaseApplication.ActiveDocument
    Set cLayerKeyStyles = doc.LayerKeyStyles
    Set dbPref = doc.Preferences
    ' Sets the layer key style to the current layer standard
    Set layerKeyStyle = cLayerKeyStyles.Item(dbPref.LayerStandard)

    Dim layer As AcadLayer
    Set layer = layerKeyStyle.GenerateLayer("WALL")
    MsgBox "Layer Key WALL" & vbInformation, "Expand Example"

End Sub
GetAnchor Example

Sub Example_GetAnchor()

' This example will get the anchor attached to an AEC Entity.

Dim ent As AcadEntity
Dim geo As AecGeo
Dim pt As Variant
Dim anchor As AecAnchor

ThisDrawing.Utility.GetEntity ent, pt, "Selected anchored object"
If TypeOf ent Is AecGeo Then
    Set geo = ent
    Set anchor = geo.GetAnchor
    If anchor Is Nothing Then
        MsgBox "AEC Entity is not anchored", vbInformation, "GetAnchor Example"
    Else
        MsgBox "AEC Entity is anchored with an " & anchor.ObjectName & " object", vbInformation, "GetAnchor Example"
    End If
Else
    MsgBox "AEC Entity not selected", vbInformation, "GetAnchor Example"
End If

End Sub
Sub Example_GetBoundingBox()
    ' This example creates a line in model space. It then finds the
    ' bounding box for the line and displays the corners of the box.

    Dim startPoint(0 To 2) As Double
    Dim endPoint(0 To 2) As Double
    Dim lineObj As AcadLine

    ' Create the Line object in model space
    startPoint(0) = 2#: startPoint(1) = 2#: startPoint(2) = 0#
    endPoint(0) = 4#: endPoint(1) = 4#: endPoint(2) = 0#
    Set lineObj = ThisDrawing.ModelSpace.AddLine(startPoint, endPoint)
    ZoomAll

    Dim minExt As Variant
    Dim maxExt As Variant

    ' Return the bounding box for the line and return the minimum
    ' and maximum extents of the box in the minExt and maxExt variables.
    lineObj.GetBoundingBox minExt, maxExt

    ' Print the min and max extents
    MsgBox "The extents of the bounding box for the line are:" & vbCrLf _
        & "Min Extent: " & minExt(0) & "," & minExt(1) & "," & minExt(2) _
        & vbCrLf & "Max Extent: " & maxExt(0) & "," & maxExt(1) & "," & maxExt(2), vbInformation, "GetBoundingBox Example"

End Sub
GetExtensionDictionary Example

Sub Example_GetExtensionDictionary()
    ' This example creates a Circle object in model space and
    ' adds a new Extension Dictionary to the Circle object

    Dim EDictionary As AcadDictionary
    Dim circleObj As AcadCircle
    Dim centerPoint(0 To 2) As Double
    Dim radius As Double

    ' Define the Circle object
    centerPoint(0) = 0: centerPoint(1) = 0: centerPoint(2) = 0
    radius = 5#

    ' Create the Circle object in model space
    Set circleObj = ThisDrawing.ModelSpace.AddCircle(centerPoint, radius)

    ThisDrawing.Application.ZoomAll

    MsgBox "Circle object Extension Dictionary state before: " & circleObj.HasExtensionDictionary

    ' Create an Extension Dictionary for the new Circle
    Set EDictionary = circleObj.GetExtensionDictionary

    MsgBox "Circle object Extension Dictionary state after: " & circleObj.HasExtensionDictionary
End Sub
Sub Example_GetXData()
' This example creates a line and attaches extended data to that line.

' Create the line
Dim lineObj As AcadLine
Dim startPt(0 To 2) As Double, endPt(0 To 2) As Double
startPt(0) = 1#: startPt(1) = 1#: startPt(2) = 0#
endPt(0) = 5#: endPt(1) = 5#: endPt(2) = 0#
Set lineObj = ThisDrawing.ModelSpace.AddLine(startPt, endPt)
ZoomAll

' Initialize all the xdata values. Note that first data in the list should be
' application name and first datatype code should be 1001
Dim DataType(0 To 9) As Integer
Dim Data(0 To 9) As Variant
Dim reals3(0 To 2) As Double
Dim worldPos(0 To 2) As Double

DataType(0) = 1001: Data(0) = "Test_Application"
DataType(1) = 1000: Data(1) = "This is a test for xdata"

DataType(2) = 1003: Data(2) = "0" ' layer
DataType(3) = 1040: Data(3) = 1.23479137438413E+40 ' real
DataType(4) = 1041: Data(4) = 1237324938 ' distance
DataType(5) = 1070: Data(5) = 32767 ' 16 bit Integer
DataType(6) = 1071: Data(6) = 32767 ' 32 bit Integer
DataType(7) = 1042: Data(7) = 10 ' scaleFactor

reals3(0) = -2.95: reals3(1) = 100: reals3(2) = -20
DataType(8) = 1010: Data(8) = reals3 ' real

worldPos(0) = 4: worldPos(1) = 400.99999999: worldPos(2) = 2.798989
DataType(9) = 1011: Data(9) = worldPos ' world space position

' Attach the xdata to the line
lineObj.SetXData DataType, Data
' Return the xdata for the line
Dim xdataOut As Variant
Dim xtypeOut As Variant
lineObj.GetXData "", xtypeOut, xdataOut

End Sub
Sub Example_Groups()
    ' This example finds the current Groups collection and
    ' adds a new group to that collection.

    Dim groupColl As AcadGroups
    Set groupColl = ThisDrawing.Groups

    ' Create a dimension style named "TEST" in current drawing
    Dim testGroup As AcadGroup
    Set testGroup = groupColl.Add("TEST")
    MsgBox "A new group called " & testGroup.name & " has been added to the Groups collection.", vbInformation, "Groups Example"
End Sub
Handle Example

Sub Example_Handle()
   ' This example creates several objects in model space.
   ' It then iterates through model space and displays the
   ' Handle for each object found.
   
   ' Create a Ray object in model space
   Dim rayObj As AcadRay
   Dim basePoint(0 To 2) As Double
   Dim SecondPoint(0 To 2) As Double
   basePoint(0) = 3#: basePoint(1) = 3#: basePoint(2) = 0#
   SecondPoint(0) = 1#: SecondPoint(1) = 3#: SecondPoint(2) = 0#
   Set rayObj = ThisDrawing.ModelSpace.AddRay(basePoint, SecondPoint)
   
   ' Create a polyline object in model space
   Dim plineObj As AcadLWPolyline
   Dim points(0 To 5) As Double
   points(0) = 3: points(1) = 7
   points(2) = 9: points(3) = 2
   points(4) = 3: points(5) = 5
   Set plineObj = ThisDrawing.ModelSpace.AddLightWeightPolyline(points)
   plineObj.Closed = True
   
   ' Create a line object in model space
   Dim lineObj As AcadLine
   Dim startPoint(0 To 2) As Double
   Dim endPoint(0 To 2) As Double
   startPoint(0) = 0: startPoint(1) = 0: startPoint(2) = 0
   endPoint(0) = 2: endPoint(1) = 2: endPoint(2) = 0
   Set lineObj = ThisDrawing.ModelSpace.AddLine(startPoint, endPoint)
   
   ' Create a circle object in model space
   Dim circObj As AcadCircle
   Dim centerPt(0 To 2) As Double
   Dim radius As Double
   centerPt(0) = 5: centerPt(1) = 3: centerPt(2) = 0
   radius = 3
Set circObj = ThisDrawing.ModelSpace.AddCircle(centerPt, radius)

' Create an ellipse object in model space
Dim ellObj As AcadEllipse
Dim majAxis(0 To 2) As Double
Dim center(0 To 2) As Double
Dim radRatio As Double
center(0) = 5#: center(1) = 5#: center(2) = 0#
majAxis(0) = 10: majAxis(1) = 20#: majAxis(2) = 0#
radRatio = 0.3
Set ellObj = ThisDrawing.ModelSpace.AddEllipse(center, majAxis, radRatio)

ZoomAll

' Iterate through the model space collection and display
' the handle of each entity found.

Dim entHandle As String
Dim entry As AcadEntity
For Each entry In ThisDrawing.ModelSpace
    entHandle = entry.handle
    entry.Highlight (True)
    MsgBox "The handle of this object is " & entHandle, vbInformation, "Handle Example"
    entry.Highlight (False)
Next
End Sub
HandleToObject Example

Sub Example_HandleToObject()
    ' This example creates a spline in model space. It then returns the
    ' handle for the spline. The spline is returned from the handle, and
    ' then colored.

    ' Create the spline
    Dim splineObj As AcadSpline
    Dim startTan(0 To 2) As Double
    Dim endTan(0 To 2) As Double
    Dim fitPoints(0 To 8) As Double

    startTan(0) = 0.5: startTan(1) = 0.5: startTan(2) = 0
    endTan(0) = 0.5: endTan(1) = 0.5: endTan(2) = 0
    fitPoints(0) = 1: fitPoints(1) = 1: fitPoints(2) = 0
    fitPoints(3) = 5: fitPoints(4) = 5: fitPoints(5) = 0
    fitPoints(6) = 10: fitPoints(7) = 0: fitPoints(8) = 0
    Set splineObj = ThisDrawing.ModelSpace.AddSpline(fitPoints, startTan, endTan)

    ZoomAll

    ' Find the handle of the spline
    Dim handle As Integer
    handle = splineObj.handle
    MsgBox "The handle of the Spline is: " & splineObj.handle, , "HandleToObject Example"

    ' Find an object from a given handle
    Dim tempObj As AcadObject
    Set tempObj = ThisDrawing.HandleToObject(handle)

    ' Now use the newly initialized object variable to color the object red
    tempObj.Color = acRed

    ThisDrawing.Regen True
    MsgBox "The Spline is now red.", , "HandleToObject Example"

End Sub
Sub Example_Has()

' This example will check the wall styles collection for a given wall style.

Dim doc As AecArchBaseDocument
Set doc = AecArchBaseApplication.ActiveDocument
Dim cWallStyles As AecWallStyles
Set cWallStyles = doc.WallStyles
Dim styleName As String
styleName = "Brick"
If cWallStyles.Has(styleName) Then
    MsgBox "Wall style: ' " & styleName & " ' in drawing", vbInformation, "Has Example" Else
    MsgBox "Wall style: ' " & styleName & " ' not in drawing", vbInformation, "Has Example"
End If

End Sub
Sub Example_HasExtensionDictionary()
    ' This example will iterate through each object in the current drawing and
    ' determine if that object has an associated Extension Dictionary
    Dim DrawingObject As AcadObject
    Dim ExtensionDictionaryResults As String

    ' Make sure this drawing contains objects before we continue
    If ThisDrawing.ModelSpace.count = 0 Then
        MsgBox "There are no objects in the current drawing."
        Exit Sub
    End If

    For Each DrawingObject In ThisDrawing.ModelSpace
        ' Determine if object contains Extension Dictionary
        Select Case DrawingObject.HasExtensionDictionary
            Case True
                ExtensionDictionaryResults = ExtensionDictionaryResults & DrawingObject.ObjectName & " has an associated Extension Dictionary"
            Case False
                ExtensionDictionaryResults = ExtensionDictionaryResults & DrawingObject.ObjectName & " does not have an associated Extension Dictionary"
        End Select
    Next

    MsgBox ExtensionDictionaryResults
End Sub
Height Example

Examples:

1 AecClipVol
1 AecLayoutGrid3D
1 AecMassElement

Sub Example_Height_AecClipVol()

'This example shows the height of the upper cut plane above the
'building elevation line position

    Dim object As Object
    Dim clip As AecClipVol
    Dim count As Integer

    'initialize
    count = 0

    For Each object In ThisDrawing.ModelSpace
        If TypeOf object Is AecClipVol Then
            count = count + 1
            Set clip = object
            MsgBox "ClipVol " & count & " Height is: " & clip.Height, vbInformation
        End If
    Next

    If count = 0 Then
        MsgBox "No ClipVol Present in Drawing", vbInformation, "Height Example"
    End If

End Sub
Sub Example_Height_AecLayoutGrid3D()

Dim object As Object Dim grid As AecLayoutGrid3D Dim count As Integer 'initialize count = 0 For Each object In ThisDrawing.ModelSpace If TypeOf object Is AecLayoutGrid3D Then count = count + 1 Set grid = object MsgBox "3D Grid" & count & " Height is: " & grid.Height, vbInformation, "Height Example" End If Next If count = 0 Then MsgBox "No 3D Layout Grids Present in Drawing", vbInformation, "Height Example" End If

End Sub

Sub Example_Height_AecMassElement()

' This example shows the size of the mass element in its relative Z direction

Dim object As Object
Dim mass As AecMassElement
Dim count As Integer

'initialize
count = 0

For Each object In ThisDrawing.ModelSpace

    If TypeOf object Is AecMassElement Then
        count = count + 1
        Set mass = object
        MsgBox "Mass Element " & count & " Height is: " & mass.Height, vbInformation, "Height Example"
    End If

Next

If count = 0 Then
    MsgBox "No Mass Elements Present in Drawing", vbInformation, "Height Example" End If
End Sub
Highlight Example

Sub Example_Highlight()
    ' This example creates several objects in model space.
    ' It then iterates through model space and highlights each
    ' object and displays it's Entity Name.

    ' Create a Ray object in model space
    Dim rayObj As AcadRay
    Dim basePoint(0 To 2) As Double
    Dim SecondPoint(0 To 2) As Double
    basePoint(0) = 3#: basePoint(1) = 3#: basePoint(2) = 0#
    SecondPoint(0) = 1#: SecondPoint(1) = 3#: SecondPoint(2) = 0#
    Set rayObj = ThisDrawing.ModelSpace.AddRay(basePoint, SecondPoint)

    ' Create a polyline object in model space
    Dim plineObj As AcadLWPolyline
    Dim points(0 To 5) As Double
    points(0) = 3: points(1) = 7
    points(2) = 9: points(3) = 2
    points(4) = 3: points(5) = 5
    Set plineObj = ThisDrawing.ModelSpace.AddLightWeightPolyline(points)
    plineObj.Closed = True

    ' Create a line object in model space
    Dim lineObj As AcadLine
    Dim startPoint(0 To 2) As Double
    Dim endPoint(0 To 2) As Double
    startPoint(0) = 0: startPoint(1) = 0: startPoint(2) = 0
    endPoint(0) = 2: endPoint(1) = 2: endPoint(2) = 0
    Set lineObj = ThisDrawing.ModelSpace.AddLine(startPoint, endPoint)

    ' Create a circle object in model space
    Dim circObj As AcadCircle
    Dim centerPt(0 To 2) As Double
    Dim radius As Double
    centerPt(0) = 20: centerPt(1) = 30: centerPt(2) = 0
    radius = 3
Set circObj = ThisDrawing.ModelSpace.AddCircle(centerPt, radius)

' Create an ellipse object in model space
Dim ellObj As AcadEllipse
Dim majAxis(0 To 2) As Double
Dim center(0 To 2) As Double
Dim radRatio As Double
center(0) = 5#: center(1) = 5#: center(2) = 0#
majAxis(0) = 10: majAxis(1) = 20#: majAxis(2) = 0#
radRatio = 0.3
Set ellObj = ThisDrawing.ModelSpace.AddEllipse(center, majAxis, radRatio)

ZoomAll

' Iterate through the model space collection and
' highlight each entity.

Dim entName As String
Dim entry As AcadEntity
For Each entry In ThisDrawing.ModelSpace
    entName = entry.ObjectName
    entry.Highlight (True)
    MsgBox "The name of this object is " & entName, vbInformation, "Highlight Example"
    entry.Highlight (False)
Next
End Sub
Sub Example_HyperLinks()
  ' This example creates a Circle object in model space and
  ' adds a new Hyperlink to its Hyperlink collection

  Dim Hyperlinks As AcadHyperlinks
  Dim Hyperlink As AcadHyperlink
  Dim circleObj As AcadCircle
  Dim centerPoint(0 To 2) As Double
  Dim radius As Double
  Dim HLLList As String

  ' Define the Circle object
  centerPoint(0) = 0: centerPoint(1) = 0: centerPoint(2) = 0
  radius = 5#

  ' Create the Circle object in model space
  Set circleObj = ThisDrawing.ModelSpace.AddCircle(centerPoint, radius)

  ThisDrawing.Application.ZoomAll

  ' Get reference to the Circle's Hyperlinks collection
  Set Hyperlinks = circleObj.Hyperlinks

  ' Add a new Hyperlink complete with all properties
  Set Hyperlink = Hyperlinks.Add("AutoDesk")
  Hyperlink.URL = "www.autodesk.com"
  Hyperlink.URLDescription = "Autodesk Main Site"
  Hyperlink.URLNamedLocation = "MY_LOCATION"

  ' Read and display a list of existing Hyperlinks and
  ' their properties for this object
  For Each Hyperlink In Hyperlinks
    HLLList = HLLList & "____________________________________" & vbCrLf
    HLLList = HLLList & "URL: " & Hyperlink.URL & vbCrLf
    HLLList = HLLList & "URL Description: " & Hyperlink.URLDescription & vbCrLf
    HLLList = HLLList & "URL Named Location: " & Hyperlink.URLNamedLocation & vbCrLf
  Next
Next

MsgBox "The circle has " & Hyperlinks.count & " Hyperlink: " & vbCrLf & HLList
End Sub
Sub Example_ImportFreeForm()

' This example imports a previously exported boundary representation
' of a mass element and applies it to a free-form mass element.
' It prompts users to specify the location of the new mass element.
' You can run this code in conjunction with Example_ExportFreeForm
' (run Example_ExportFreeForm first).

Dim massElement As AecMassElement
Dim pt As Variant
Dim center_at_origin As Boolean

' Create a new mass element
Set massElement = ThisDrawing.ModelSpace.AddCustomObject("AecMassElé

' Set the mass element type to "free-form"
massElement.Type = aecMassElementTypeFreeForm

'Select a location for the mass element
pt = ThisDrawing.Utility.GetPoint(, "Select the insertion point:")
If Err.Number  0 Then
   MsgBox ("error when getting a point." & vbCrLf)
   Exit Sub
End If

massElement.Location = pt

' Use mass element Location as insertion point.
center_at_origin = True

' Import the boundary representation.
massElement.ImportFreeForm "c:\temp\freeform-massElement"

End Sub
Index Example

Examples:

- AecLayerKey
- AecLayerOverrideSetting
- AecLayoutCurveNode
- AecLayoutGrid2DNode
- AecLayoutGrid2DXNode
- AecLayoutGrid2DYNode
- AecLayoutGrid3DNode
- AecLayoutGrid3DXNode
- AecLayoutGrid3DYNode
- AecLayoutGrid3DZNode
- AecRing
- AecViewBlock

Sub Example_Index_AecLayerKey()

'This example shows the index of the layer generated by the key

Dim db As New AecBaseDatabase
Dim layerKey As AecLayerKey

db.Init ThisDrawing.Database

Set layerKey = db.LayerKeyStyles.Item(0).Keys.Item(0)
MsgBox "Layer Index: " & layerKey.Index, vbInformation, "Index Example"
Sub Example_Index_AecLayerOverrideSetting()

' This example shows the index of the override setting

Dim db As New AecBaseDatabase
Dim setting As AecLayerOverrideSetting

db.Init ThisDrawing.Database

Set setting = db.LayerKeyStyles.Item(0).overrideSettings.Item(0)
MsgBox "Setting Index: " & setting.Index, vbInformation, "Index Example"

End Sub

Sub Example_Index_AecLayoutCurveNode()

' This example displays the color of the nodes owner layout curve

Dim obj As Object
Dim pt As Variant
Dim layoutCurve As AecLayoutCurve
Dim node As AecLayoutCurveNode

ThisDrawing.Utility.GetEntity obj, pt, "Select a Layout Curve"

If TypeOf obj Is AecLayoutCurve Then
    Set layoutCurve = obj
    Set node = layoutCurve.nodes.Item(0)
    MsgBox "Color is: " & node.layoutCurve.Color, vbInformation, "LayoutCurve Example"
Else
    MsgBox "Not a Layout Curve", vbExclamation, "LayoutCurve Example"
End If
Sub Example_Index_AecLayoutGrid2DNode()

    'This example displays the index of a layout node

    Dim obj As Object
    Dim pt As Variant
    Dim grid As AecLayoutGrid2D
    Dim node As AecLayoutGrid2DNode

    ThisDrawing.Utility.GetEntity obj, pt, "Select a 2D Layout Grid"

    If TypeOf obj Is AecLayoutGrid2D Then
        Set grid = obj
        Set node = grid.XNodes.Item(0)
        MsgBox "Node Index is: " & node.Index, vbInformation, "Index Example"
    Else
        MsgBox "Not a 2D Layout Grid", vbExclamation, "Index Example"
    End If

End Sub

Sub Example_Index_AecLayoutGrid2DXNode()

    'This example displays the index of a layout Xnode

    Dim obj As Object
    Dim pt As Variant
    Dim grid As AecLayoutGrid2D
    Dim node As AecLayoutGrid2DNode

    ThisDrawing.Utility.GetEntity obj, pt, "Select a 2D Layout Grid"
If TypeOf obj Is AecLayoutGrid2D Then
    Set grid = obj
    Set node = grid.XNodes.Item(0)
    MsgBox "XNode Index is: " & node.Index, vbInformation, "Index Example"
Else
    MsgBox "Not a 2D Layout Grid", vbExclamation, "Index Example"
End If

End Sub

Sub Example_Index_AecLayoutGrid2DYNode()

    'This example displays the index of a layout YNode

    Dim obj As Object
    Dim pt As Variant
    Dim grid As AecLayoutGrid2D
    Dim node As AecLayoutGrid2DNode

    ThisDrawing.Utility.GetEntity obj, pt, "Select a 2D Layout Grid"

    If TypeOf obj Is AecLayoutGrid2D Then
        Set grid = obj
        Set node = grid.YNodes.Item(0)
        MsgBox "YNode Index is: " & node.Index, vbInformation, "Index Example"
    Else
        MsgBox "Not a 2D Layout Grid", vbExclamation, "Index Example"
    End If

End Sub

Sub Example_Index_AecLayoutGrid3DNode()

    'This example displays the index of a layout node
Dim obj As Object  
Dim pt As Variant  
Dim grid As AecLayoutGrid3D  
Dim node As AecLayoutGrid3DNode

ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

If TypeOf obj Is AecLayoutGrid3D Then  
    Set grid = obj  
    Set node = grid.XNodes.Item(0)  
    MsgBox "Node Index is: " & node.Index, vbInformation, "Index Example"  
Else  
    MsgBox "Not a 3D Layout Grid", vbExclamation, "Index Example"  
End If

End Sub

Sub Example_Index_AecLayoutGrid3DXNode()

    'This example displays the index of a layout Xnode

    Dim obj As Object  
    Dim pt As Variant  
    Dim grid As AecLayoutGrid3D  
    Dim node As AecLayoutGrid3DNode

    ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

    If TypeOf obj Is AecLayoutGrid3D Then  
        Set grid = obj  
        Set node = grid.XNodes.Item(0)  
        MsgBox "XNode Index is: " & node.Index, vbInformation, "Index Example'  
    Else  
        MsgBox "Not a 3D Layout Grid", vbExclamation, "Index Example"  
    End If

End Sub
Sub Example_Index_AecLayoutGrid3DYNode()

'This example displays the index of a layout Ynode

Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid3D
Dim node As AecLayoutGrid3DNode

ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

If TypeOf obj Is AecLayoutGrid3D Then
    Set grid = obj
    Set node = grid.YNodes.Item(0)
    MsgBox "YNode Index is: " & node.Index, vbInformation, "Index Example"
Else
    MsgBox "Not a 3D Layout Grid", vbExclamation, "Index Example"
End If

End Sub

Sub Example_Index_AecLayoutGrid3DZNode()

'This example displays the index of a layout Znode

Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid3D
Dim node As AecLayoutGrid3DNode

ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

If TypeOf obj Is AecLayoutGrid3D Then
    Set grid = obj

Set node = grid.ZNodes.Item(0)
MsgBox "ZNode Index is: " & node.Index, vbInformation, "Index Example"
Else
    MsgBox "Not a 3D Layout Grid", vbExclamation, "Index Example"
End If

End Sub

Sub Example_Index_AecRing()

' This example will display the index of the ring

Dim db As New AecBaseDatabase
Dim ring As AecRing

db.Init ThisDrawing.Database

Set ring = db.ProfileStyles.Item(0).Profile.rings.Item(0)
MsgBox "Ring Index: " & ring.Index, vbInformation, "Index Example"

End Sub

Sub Example_Index_AecViewBlock()

' This example shows the index of the first viewblock of a multiviewblock

Dim obj As Object
Dim pt As Variant
Dim blockRef As AecMVBlockRef
Dim viewBlocks As AecViewBlocks

ThisDrawing.Utility.GetEntity obj, pt, "Select a Multiview Block"
If TypeOf obj Is AecMVBlockRef Then
    Set blockRef = obj
Set viewBlocks = blockRef.viewBlocks
MsgBox "Index of View Block 1: " & viewBlocks.Item(0).Index, vbInformation
Else
    MsgBox "Not a Multiview Block", vbInformation, "Index Example"
End If

End Sub
Sub Example_Init()

    ' This example initializes the application object, using
    ' the current drawing, then lists the number of objects in
    ' the drawing's model space.

    Dim app As New AecBaseApplication
    Dim doc As AecBaseDocument

    Dim nbr_of_objects As Integer
    Dim msg As String

    ' Initialize the application object.
    app.Init ThisDrawing.Application

    ' Access the current drawing database.
    Set doc = app.ActiveDocument

    ' Get the number of objects in the drawing's model space.
    nbr_of_objects = doc.ModelSpace.Count

    msg = "There are " & nbr_of_objects & " objects in this drawing's model space.
    MsgBox msg, vbInformation, "Init Example"

End Sub
**Intersect Example**

Sub Example_Intersect()

' This example will add make two profiles from rings, and intersect the first with the second

On Error Resume Next
Dim pointList1(0 To 9) As Double
Dim pointList2(0 To 7) As Double
pointList1(0) = 0: pointList1(1) = 0
pointList1(2) = 1: pointList1(3) = 0
pointList1(4) = 1: pointList1(5) = 1
pointList1(6) = 0: pointList1(7) = 1
pointList1(8) = 0: pointList1(9) = 0

pointList2(0) = 0.5: pointList2(1) = 0.5
pointList2(2) = 0.067: pointList2(3) = -0.25
pointList2(4) = 0.933: pointList2(5) = -0.25
pointList2(6) = 0.5: pointList2(7) = 0.5

Dim ring1 As AecRing
Dim ring2 As AecRing
Dim profile1 As New AecProfile
Dim profile2 As New AecProfile

Dim doc As AecArchBaseDocument
Set doc = AecArchBaseApplication.ActiveDocument
Dim cprofiles As AecProfileStyles
Dim profileStyle As AecProfileStyle

Set cprofiles = doc.ProfileStyles
Set profileStyle = cprofiles.Item("IntersectRing")
If profileStyle Is Nothing Then
    Set profileStyle = cprofiles.Add("IntersectRing")
End If
Set ring1 = profile1.Rings.Add
Set ring2 = profile2.Rings.Add
ring1.FromPoints (pointList1)
ring2.FromPoints (pointList2)
profile1.Intersect profile2

*Set* profileStyle.profile = profile1

*End Sub*
Sub Example_IntersectWith()
    ' This example creates a line and circle and finds the points at
    ' which they intersect.
    
    ' Create the line
    Dim lineObj As AcadLine
    Dim startPoint(0 To 2) As Double
    Dim endPoint(0 To 2) As Double
    startPoint(0) = 1: startPoint(1) = 1: startPoint(2) = 0
    endPoint(0) = 5: endPoint(1) = 5: endPoint(2) = 0
    Set lineObj = ThisDrawing.ModelSpace.AddLine(startPoint, endPoint)
    
    ' Create the circle
    Dim circleObj As AcadCircle
    Dim centerPoint(0 To 2) As Double
    Dim radius As Double
    centerPoint(0) = 3: centerPoint(1) = 3: centerPoint(2) = 0
    radius = 1
    Set circleObj = ThisDrawing.ModelSpace.AddCircle(centerPoint, radius)
    ZoomAll
    
    ' Find the intersection points between the line and the circle
    Dim intPoints As Variant
    intPoints = lineObj.IntersectWith(circleObj, acExtendNone)
    
    ' Print all the intersection points
    Dim I As Integer, j As Integer, k As Integer
    Dim str As String
    If VarType(intPoints) <> vbEmpty Then
        For I = LBound(intPoints) To UBound(intPoints)
            str = "Intersection Point["] & k & "] is: " & intPoints(j) & "," & intPoints(j + 1)
            MsgBox str, , "IntersectWith Example"
            str = ""
            I = I + 2
            j = j + 3
            k = k + 1
        Next I
    End If
End Sub
Next
End If
End Sub
**Item Example**

Sub Example_Item()

' This example lists the number of keys in the Standard
' layer key styles collection, and then lists the name of
' each layer key styles collection in the drawing. The example
' uses both the string and integer methods of specifying the
' index of an Item.

Dim app As New AecBaseApplication
Dim doc As AecBaseDocument
Dim layerkeystyles As AecLayerKeyStyles
Dim keystyle As AecLayerKeyStyle
Dim msg As String

app.Init ThisDrawing.Application
Set doc = app.ActiveDocument

' Get the drawing's collection of layer key styles.
Set layerkeystyles = doc.layerkeystyles
msg = "Number of Layer Key Styles in this drawing is: " & layerkeystyles.Count & vbCrLf

' Get the standard layer key style, if it exists (it should!)
On Error Resume Next
Set keystyle = layerkeystyles.Item("Standard")
Err.Clear
Else
End If
On Error GoTo 0

' Loop through the layer key styles collection, listing the name
' of each layer key style in the collection.
msg = msg & vbCrLf & "The layer key styles in this drawing are:" & vbCrLf
For i = 0 To (layerkeystyles.Count - 1)
    msg = msg & "   " & layerkeystyles.Item(i).Name & vbCrLf
Next

MsgBox msg, vbInformation, "Item Example"

End Sub
Keys Example

Sub Example_Keys()

'This example shows the how to get the layer key from the layer key style.

Dim doc As AecArchBaseDocument
Dim dbPref As AecArchBaseDatabasePreferences
Dim cLayerKeyStyles As AecLayerKeyStyles
Dim layerKeyStyle As AecLayerKeyStyle
Dim cLayerKeys As AecLayerKeys
Dim layerKey As AecLayerKey

Set doc = AecArchBaseApplication.ActiveDocument
Set cLayerKeyStyles = doc.LayerKeyStyles
Set dbPref = doc.Preferences
'Sets the layer key style to the current layer standard
Set layerKeyStyle = cLayerKeyStyles.Item(dbPref.LayerStandard)
Set cLayerKeys = layerKeyStyle.Keys

For Each layerKey In cLayerKeys
    Debug.Print layerKey.Name
    Debug.Print " Color      - " & layerKey.Color
    Debug.Print " Layer      - " & layerKey.Layer
    Debug.Print " LineType   - " & layerKey.Linetype
    Debug.Print " Lineweight - " & layerKey.Lineweight
    Debug.Print " Plotstyle - " & layerKey.PlotStyleName
    Debug.Print " Plottable - " & layerKey.Plottable
    Debug.Print " Removable - " & layerKey.Removeable
Next

< PRE class= Code >

End Sub
Sub Example_Layer()

' This example lists the layer keys in the layer key style
' of the document's standard layer.

Dim app As New AecBaseApplication
Dim doc As AecBaseDocument
Dim dbPref As AecBaseDatabasePreferences
Dim cLayerKeyStyles As AecLayerKeyStyles
Dim layerKeyStyle As AecLayerKeyStyle
Dim cLayerKeys As AecLayerKeys
Dim layerKey As AecLayerKey

Dim msg As String

' Initialize the application object and access the current drawing.
app.Init ThisDrawing.Application
Set doc = app.ActiveDocument

' Get the drawing's collection of layer key styles.
Set cLayerKeyStyles = doc.layerkeystyles

' Get the preferences object.
Set dbPref = doc.Preferences

' Identify the layer standard.
msg = "Layer standard is " & dbPref.LayerStandard _
   & ". It contains the following layer keys:" & vbCrLf

' Set the layer key style to the current layer standard.
Set layerKeyStyle = cLayerKeyStyles.Item(dbPref.LayerStandard)

' Get the collection of layer keys in the style.
Set cLayerKeys = layerKeyStyle.Keys

' Loop through the collection and list some properties of each key.
For Each layerKey In cLayerKeys
    msg = msg & " " & layerKey.Name & ":" & vbCrLf
    msg = msg & " Color   - " & layerKey.Color & vbCrLf
    msg = msg & " Layer   - " & layerKey.Layer & vbCrLf
    msg = msg & " Linetype - " & layerKey.Linetype & vbCrLf
    msg = msg & " Lineweight - " & layerKey.Lineweight & vbCrLf
    msg = msg & " Plotstyle - " & layerKey.PlotStyleName & vbCrLf
Next

MsgBox msg, vbInformation, "Layer Example"

End Sub
LayerFile Example

Sub Example_LayerFile()

' This example displays the LayerFile setting for the current drawing.

Dim dbPref As AecArchBaseDatabasePreferences

Set dbPref = AecArchBaseApplication.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 Example"

End Sub
LayerKeyStyle Example

Examples:

- `AecLayerKey`
- `AecLayerKeys`
- `AecLayerOverrideSetting`
- `AecLayerOverrideSettings`

---

Sub Example_LayerKeyStyle_AecLayerKey()

'This example shows the layer style generated by the key

```vba
Dim db As New AecBaseDatabase
Dim layerKey As AecLayerKey

db.Init ThisDrawing.Database

Set layerKey = db.LayerKeyStyles.Item(0).Keys.Item(0)
MsgBox "Layer Style: " & layerKey.layerKeyStyle.Name, vbInformation, "LayerKeyStyle Example"

End Sub
```

---

Sub Example_LayerKeyStyle_AecLayerKeys()

'This example shows the name of the owner layer key style

```vba
Dim db As New AecBaseDatabase
Dim layerKeys As AecLayerKeys

db.Init ThisDrawing.Database

Set layerKeys = db.LayerKeyStyles.Item(0).Keys
```
MsgBox "Layer Keys Owner Name: " & layerKeys.layerKeyStyle.Name, vbInformation

End Sub

Sub Example_LayerKeyStyle_AecLayerOverrideSetting()

' This example shows the name of the owner of the override setting

Dim db As New AecBaseDatabase
Dim setting As AecLayerOverrideSetting

db.Init ThisDrawing.Database

Set setting = db.LayerKeyStyles.Item(0).overrideSettings.Item(0)
MsgBox "Setting Owners Name: " & setting.layerKeyStyle.Name, vbInformation

End Sub

Sub Example_LayerKeyStyle_AecLayerOverrideSettings()

Dim db As New AecBaseDatabase
Dim overrideSettings As AecLayerOverrideSettings

db.Init ThisDrawing.Database

Set overrideSettings = db.LayerKeyStyles.Item(0).overrideSettings
MsgBox "Layer Override Settings owner name: " & overrideSettings.layerKeyStyle.Name, vbInformation

End Sub
Sub Example_LayerKeyStyles()

    ' This example will display the number of Layer Key Styles in the current drawing.

    Dim doc As AecArchBaseDocument
    Dim cLayerKeyStyles As AecLayerKeyStyles

    Set doc = AecArchBaseApplication.ActiveDocument
    Set cLayerKeyStyles = doc.LayerKeyStyles
    MsgBox "Number of Layer Key Styles in this drawing is: " & cLayerKeyStyles.Count,

End Sub
Layers Example

Sub Example_Layers()
    ' This example finds the current Layers collection and
    ' adds a new layer to that collection.

    Dim layerColl As AcadLayers
    Set layerColl = ThisDrawing.Layers

    ' Create a dimension style named "TEST" in current drawing
    Dim testlayer As AcadLayer
    Set testlayer = layerColl.Add("TEST")
    MsgBox "A new layer called " & testlayer.name & " has been added to the Lay

End Sub
Sub Example_LayerStandard()

' This example displays the LayerStandard setting for the current drawing.

Dim dbPref As AecArchBaseDatabasePreferences

Set dbPref = AecArchBaseApplication.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
**LayoutCurve Example**

**Examples:**

1. AecLayoutCurveNode
2. AecLayoutCurveNodes

---

```vba
Sub Example_LayoutCurve_AecLayoutCurveNode()
    'This example displays the color of the nodes owner layout curve

    Dim obj As Object
    Dim pt As Variant
    Dim layoutCurve As AecLayoutCurve
    Dim node As AecLayoutCurveNode

    ThisDrawing.Utility.GetEntity obj, pt, "Select a Layout Curve"

    If TypeOf obj Is AecLayoutCurve Then
        Set layoutCurve = obj
        Set node = layoutCurve.nodes.Item(0)
        MsgBox "Color is: " & node.layoutCurve.Color, vbInformation, "LayoutCurve Example"
    Else
        MsgBox "Not a Layout Curve", vbExclamation, "LayoutCurve Example"
    End If

End Sub
```

---

```vba
Sub Example_LayoutCurve_AecLayoutCurveNodes()
    'This example displays the color of the node collections owner curve

    Dim obj As Object
    Dim pt As Variant
```

---
Dim layoutCurve As AecLayoutCurve
Dim nodes As AecLayoutCurveNodes

ThisDrawing.Utility.GetEntity obj, pt, "Select a Layout Curve"

If TypeOf obj Is AecLayoutCurve Then
    Set layoutCurve = obj
    Set nodes = layoutCurve.nodes
    MsgBox "Color is: " & nodes.layoutCurve.Color, vbInformation, "LayoutCurve Example"
Else
    MsgBox "Not a Layout Curve", vbExclamation, "LayoutCurve Example"
End If

End Sub
LayoutGrid2D Example

Examples:

- **AecLayoutGrid2DNode**
- **AecLayoutGrid2DNodes**
- **AecLayoutGrid2DXNode**
- **AecLayoutGrid2DXNodes**
- **AecLayoutGrid2DYNode**
- **AecLayoutGrid2DYNodes**

Sub Example_LayoutGrid2D_AecLayoutGrid2DNode()

'This example displays the index of a layout node

Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid2D
Dim node As AecLayoutGrid2DNode

ThisDrawing.Utility.GetEntity obj, pt, "Select a 2D Layout Grid"

If TypeOf obj Is AecLayoutGrid2D Then
    Set grid = obj
    Set node = grid.XNodes.Item(0)
    MsgBox "Node LayoutGrid's Depth is: " & node.LayoutGrid2D.Depth, vbInfo
Else
    MsgBox "Not a 2D Layout Grid", vbExclamation, "LayoutGrid2D Example"
End If

End Sub
Sub Example_LayoutGrid2D_AecLayoutGrid2DNodes()

'This example displays the index of a layout node

Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid2D
Dim nodes As AecLayoutGrid2DNodes

ThisDrawing.Utility.GetEntity obj, pt, "Select a 2D Layout Grid"

If TypeOf obj Is AecLayoutGrid2D Then
    Set grid = obj
    Set nodes = grid.XNodes
    MsgBox "Node collection owner depth is: " & nodes.LayoutGrid2D.Depth, vbExclamation, "LayoutGrid2D Example"
Else
    MsgBox "Not a 2D Layout Grid", vbExclamation, "LayoutGrid2D Example"
End If

End Sub

Sub Example_LayoutGrid2D_AecLayoutGrid2DXNode()

'This example displays the depth of the owner grid of a layout xnode

Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid2D
Dim node As AecLayoutGrid2DNode

ThisDrawing.Utility.GetEntity obj, pt, "Select a 2D Layout Grid"

If TypeOf obj Is AecLayoutGrid2D Then
    Set grid = obj
    Set node = grid.XNodes.Item(0)
    MsgBox "Owner Grid Depth is: " & node.LayoutGrid2D.Depth, vbInformation
Else

End If
MsgBox "Not a 2D Layout Grid", vbExclamation, "LayoutGrid2D Example"
End If

End Sub

Sub Example_LayoutGrid2D_AecLayoutGrid2DXNodes()

' This example displays the depth of the owner grid of a collection of layout Xn

Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid2D
Dim nodes As AecLayoutGrid2DNodes

ThisDrawing.Utility.GetEntity obj, pt, "Select a 2D Layout Grid"

If TypeOf obj Is AecLayoutGrid2D Then
    Set grid = obj
    Set nodes = grid.XNodes
    MsgBox "Owner Grid Depth is: " & nodes.LayoutGrid2D.Depth, vbInformation
Else
    MsgBox "Not a 2D Layout Grid", vbExclamation, "LayoutGrid2D Example"
End If

End Sub

Sub Example_LayoutGrid2D_AecLayoutGrid2DYNode()

' This example displays the depth of the owner grid of a layout xnode

Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid2D
Dim node As AecLayoutGrid2DNode
ThisDrawing.Utility.GetEntity obj, pt, "Select a 2D Layout Grid"

If TypeOf obj Is AecLayoutGrid2D Then
    Set grid = obj
    Set node = grid.YNodes.Item(0)
    MsgBox "Owner Grid Depth is: " & node.LayoutGrid2D.Depth, vbInformation, "LayoutGrid2D Example"
Else
    MsgBox "Not a 2D Layout Grid", vbExclamation, "LayoutGrid2D Example"
End If

End Sub

Sub Example_LayoutGrid2D_AecLayoutGrid2DYNodes()
    'This example displays the depth of the owner grid of a collection of layout Yn
    Dim obj As Object
    Dim pt As Variant
    Dim grid As AecLayoutGrid2D
    Dim nodes As AecLayoutGrid2DNodes

    ThisDrawing.Utility.GetEntity obj, pt, "Select a 2D Layout Grid"

    If TypeOf obj Is AecLayoutGrid2D Then
        Set grid = obj
        Set nodes = grid.YNodes
        MsgBox "Owner Grid Depth is: " & nodes.LayoutGrid2D.Depth, vbInformation, "LayoutGrid2D Example"
    Else
        MsgBox "Not a 2D Layout Grid", vbExclamation, "LayoutGrid2D Example"
    End If

End Sub
LayoutGrid3D Example

Examples:

- **AecLayoutGrid3DNode**
- **AecLayoutGrid3DNodes**
- **AecLayoutGrid3DXNode**
- **AecLayoutGrid3DXNodes**
- **AecLayoutGrid3DYNode**
- **AecLayoutGrid3DYNodes**
- **AecLayoutGrid3DZNode**
- **AecLayoutGrid3DZNodes**

---

Sub Example_LayoutGrid3D_AecLayoutGrid3DNode()

' This example displays the index of a layout node

Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid3D
Dim node As AecLayoutGrid3DNode

ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

If TypeOf obj Is AecLayoutGrid3D Then
    Set grid = obj
    Set node = grid.XNodes.Item(0)
    MsgBox "Node LayoutGrid's Depth is: " & node.LayoutGrid3D.Depth, vbInformation, "LayoutGrid3D Example"
Else
    MsgBox "Not a 3D Layout Grid", vbExclamation, "LayoutGrid3D Example"
End If
End Sub

Sub Example_LayoutGrid3D_AecLayoutGrid3DNodes()

' This example displays the index of a layout node

Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid3D
Dim nodes As AecLayoutGrid3DNodes

ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

If TypeOf obj Is AecLayoutGrid3D Then
    Set grid = obj
    Set nodes = grid.XNodes
    MsgBox "Node collection owner depth is: " & nodes.LayoutGrid3D.Depth,
Else
    MsgBox "Not a 3D Layout Grid", vbExclamation, "LayoutGrid3D Example"
End If

End Sub

Sub Example_LayoutGrid3D_AecLayoutGrid3DXNode()

' This example displays the depth of the owner grid of a layout xnode

Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid3D
Dim node As AecLayoutGrid3DNode

ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"
If TypeOf obj Is AecLayoutGrid3D Then
    Set grid = obj
    Set node = grid.XNodes.Item(0)
    MsgBox "Owner Grid Depth is: " & node.LayoutGrid3D.Depth, vbInformation, "LayoutGrid3D Example"
Else
    MsgBox "Not a 3D Layout Grid", vbExclamation, "LayoutGrid3D Example"
End If

Sub Example_LayoutGrid3D_AecLayoutGrid3DXNodes()

    'This example displays the depth of the owner grid of a collection of layout Xn
    Dim obj As Object
    Dim pt As Variant
    Dim grid As AecLayoutGrid3D
    Dim nodes As AecLayoutGrid3DNodes

    ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

    If TypeOf obj Is AecLayoutGrid3D Then
        Set grid = obj
        Set nodes = grid.XNodes
        MsgBox "Owner Grid Depth is: " & nodes.LayoutGrid3D.Depth, vbInformation
    Else
        MsgBox "Not a 3D Layout Grid", vbExclamation
    End If

End Sub

Sub Example_LayoutGrid3D_AecLayoutGrid3DYNode()

    'This example displays the depth of the owner grid of a layout Ynode

End Sub
Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid3D
Dim node As AecLayoutGrid3DNode

ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

If TypeOf obj Is AecLayoutGrid3D Then
  Set grid = obj
  Set node = grid.YNodes.Item(0)
  MsgBox "Owner Grid Depth is: " & node.LayoutGrid3D.Depth, vbInformation, "LayoutGrid3D Example"
Else
  MsgBox "Not a 3D Layout Grid", vbExclamation, "LayoutGrid3D Example"
End If

End Sub

Sub Example_LayoutGrid3D_AecLayoutGrid3DYNodes()

  'This example displays the depth of the owner grid of a collection of layout Yn

  Dim obj As Object
  Dim pt As Variant
  Dim grid As AecLayoutGrid3D
  Dim nodes As AecLayoutGrid3DNodes

  ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

  If TypeOf obj Is AecLayoutGrid3D Then
    Set grid = obj
    Set nodes = grid.YNodes
    MsgBox "Owner Grid Depth is: " & nodes.LayoutGrid3D.Depth, vbInformation, "LayoutGrid3D Example"
  Else
    MsgBox "Not a 3D Layout Grid", vbExclamation, "LayoutGrid3D Example"
  End If

End Sub
Sub Example_LayoutGrid3D_AecLayoutGrid3DZNode()

' This example displays the depth of the owner grid of a layout Znode

Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid3D
Dim node As AecLayoutGrid3DNode

ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

If TypeOf obj Is AecLayoutGrid3D Then
    Set grid = obj
    Set node = grid.ZNodes.Item(0)
    MsgBox "Owner Grid Depth is: " & node.LayoutGrid3D.Depth, vbInformation, "LayoutGrid3D Example"
Else
    MsgBox "Not a 3D Layout Grid", vbExclamation, "LayoutGrid3D Example"
End If

End Sub

Sub Example_LayoutGrid3D_AecLayoutGrid3DZNodes()

' This example displays the depth of the owner grid of a collection of layout Znodes

Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid3D
Dim nodes As AecLayoutGrid3DNodes

ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

If TypeOf obj Is AecLayoutGrid3D Then
    Set grid = obj

Set nodes = grid.ZNodes
MsgBox "Owner Grid Depth is: " & nodes.LayoutGrid3D.Depth, vbInformation
Else
   MsgBox "Not a 3D Layout Grid", vbExclamation, "LayoutGrid3D Example"
End If

End Sub
Sub Example_Layou(t)()
    ' This example will access the Layouts collection for the current drawing
    ' and list basic information about the Layouts in the drawing.

    Dim Layouts As AcadLayouts, Layout As ACADLayout
    Dim msg As String

    ' Get layouts collection from document object
    Set Layouts = ThisDrawing.Layouts

    msg = vbCrLf & vbCrLf ' Start with a space

    ' Get the names of every layout in this drawing
    For Each Layout In Layouts
        msg = msg & Layout.name & vbCrLf
    Next

    ' Display a list of available layouts
    MsgBox "There are " & Layouts.count & " layouts in " & _
        ThisDrawing.WindowTitle & ":" & msg
End Sub
Sub Example_LeaderExtension1()

    'This example will find the 1st Leader Extension for a bubble attached to a ' column grid

    Dim obj As AcadObject
    Dim bubble As AecMVBlockRef
    Dim anchor As AecAnchor
    Dim leaderAnchor As AecAnchorLeadEntToNode
    Dim pt As Variant

    ThisDrawing.Utility.GetEntity obj, pt, "Select bubble"
    If TypeOf obj Is AecMVBlockRef Then
        Set bubble = obj
        Set anchor = obj.GetAnchor
        If TypeOf anchor Is AecAnchorLeadEntToNode Then
            Set leaderAnchor = anchor
            MsgBox "Leader Extension1 = " & leaderAnchor.LeaderExtension1
        Else
            MsgBox "Not anchored to column grid", vbInformation, "Example LeaderExtension1"
        End If
    Else
        MsgBox "Not a bubble", vbInformation, "Example LeaderExtension1"
    End If

End Sub
LeaderExtension2 Example

Sub Example_LeaderExtension2()

    'This example will find the 2nd Leader Extension for a bubble attached to a ' column grid

    Dim obj As AcadObject
    Dim bubble As AecMVBlockRef
    Dim anchor As AecAnchor
    Dim leaderAnchor As AecAnchorLeadEntToNode
    Dim pt As Variant

    ThisDrawing.Utility.GetEntity obj, pt, "Select bubble"
    If TypeOf obj Is AecMVBlockRef Then
        Set bubble = obj
        Set anchor = obj.GetAnchor
        If TypeOf anchor Is AecAnchorLeadEntToNode Then
            Set leaderAnchor = anchor
            MsgBox "Leader Extension2 = " & leaderAnchor.LeaderExtension2
        Else
            MsgBox "Not anchored to column grid", vbInformation, "Example LeaderExtension2"
        End If
    Else
        MsgBox "Not a bubble", vbInformation, "Example LeaderExtension2"
    End If

End Sub
Sub Example_LeftOffset()

    ' This example returns the left offset of the selected object
    ' to the grid assembly.

    ' Use this example with a drawing that contains a window
    ' assembly and one or more AEC objects attached to the
    ' assembly.

    Dim ent As AcadEntity
    Dim geo As AecGeo
    Dim anchor As AecAnchor
    Dim offset As String

    On Error Resume Next            ' Handle errors in code.

    ' Prompt user to select an object.
    ThisDrawing.Utility.GetEntity ent, pt, "Select object anchored to window assembly:

    ' Make sure user selected an AEC object, and that the object
    ' is anchored to a grid assembly.
    If ent Is Nothing Then
        MsgBox "Nothing was selected.", vbExclamation, "LeftOffset Example"
    ElseIf TypeOf ent Is AecGeo Then
        Set geo = ent

        ' Get the anchor the object is attached to.
        Set anchor = geo.GetAnchor
        On Error GoTo 0
        If anchor Is Nothing Then
            MsgBox "Selected object is not anchored.", vbExclamation, "LeftOffset Example"
        ElseIf Not TypeOf anchor Is AecAnchorEntToGridAssembly Then
            MsgBox "Object is anchored, but not to a grid assembly.", vbExclamation
        Else
            MsgBox "Left offset of object: " & anchor.LeftOffset, vbInformation, "LeftOffset Example"
    End If

    ' Handle remaining errors.
    On Error GoTo 0

End Sub
End If
Else
    MsgBox "Object selected is not an AEC entity.", vbInformation, "LeftOffset"
End If

End Sub
Sub Example_Limits()
    ' This example finds the current limits for the drawing.
    ' It then changes the limits for the drawing. The grid
    ' is turned on to show the limits.
    ' Turn on the grid for the active viewport
    ThisDrawing.ActiveViewport.GridOn = True
    ThisDrawing.ActiveViewport = ThisDrawing.ActiveViewport
    ' Find the current limits
    Dim currLimits As Variant
    currLimits = ThisDrawing.Limits
    MsgBox "The current drawing limits are " & vbCrLf & & "Lower-left corner " & ThisDrawing.Limits(0) & ", " & ThisDrawing.Limits(1) & vbCrLf & & "Upper-right corner " & ThisDrawing.Limits(2) & ", " & ThisDrawing.Limits(3), , "Limits Example"
    ' Change the limits
    Dim newLimits(0 To 3) As Double
    newLimits(0) = 2#: newLimits(1) = 2#: newLimits(2) = 4#: newLimits(3) = 4#
    ThisDrawing.Limits = newLimits
    ThisDrawing.Regen acActiveViewport
    MsgBox "The new drawing limits are " & vbCrLf & & "Lower-left corner " & ThisDrawing.Limits(0) & " , " & ThisDrawing.Limits(1) & vbCrLf & & "Upper-right corner " & ThisDrawing.Limits(2) & ", " & ThisDrawing.Limits(3), , "Limits Example"
    ' Reset the drawing limits
    ThisDrawing.Limits = currLimits
    ThisDrawing.Regen acActiveViewport
    MsgBox "The drawing limits have been reset to " & vbCrLf & & "Lower-left corner " & ThisDrawing.Limits(0) & " , " & ThisDrawing.Limits(1) & vbCrLf & & "Upper-right corner " & ThisDrawing.Limits(2) & " , " & ThisDrawing.Limits(3), , "Limits Example"
End Sub
Sub Example_LinearDisplayFormat()

' This example returns the LinearDisplayFormat setting for the current drawing

Dim dbPref As AecArchBaseDatabasePreferences

Set dbPref = AecArchBaseApplication.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 AecArchBaseDatabasePreferences
    Set dbPref = AecArchBaseApplication.ActiveDocument.Preferences

    MsgBox "The current value for LinearPrecision is: " & dbPref.LinearPrecision,
            vbInformation, "LinearPrecision Example"

End Sub
Sub Example_LinearUnit()

' This example returns the LinearUnit setting for the current drawing.
Dim dbPref As AecArchBaseDatabasePreferences
Set dbPref = AecArchBaseApplication.ActiveDocument.preferences

' Convert the linear 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 aecUnitMillimeter
    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, "Linear
LineType Example

Sub Example_LineType ()

'This example shows the linetype of the layer generated by the key

Dim doc As AecArchBaseDocument
Dim dbPref As AecArchBaseDatabasePreferences
Dim cLayerKeyStyles As AecLayerKeyStyles
Dim layerKeyStyle As AecLayerKeyStyle
Dim cLayerKeys As AecLayerKeys
Dim layerKey As AecLayerKey

Set doc = AecArchBaseApplication.ActiveDocument
Set cLayerKeyStyles = doc.LayerKeyStyles
Set dbPref = doc.Preferences
' Sets the layer key style to the current layer standard
Set layerKeyStyle = cLayerKeyStyles.Item(dbPref.LayerStandard)
Set cLayerKeys = layerKeyStyle.Keys

For Each layerKey In cLayerKeys
    Debug.Print layerKey.Name
    Debug.Print " Color   - " & layerKey.Color
    Debug.Print " Layer   - " & layerKey.Layer
    Debug.Print " LineType - " & layerKey.Linetype
    Debug.Print " Lineweight - " & layerKey.Lineweight
    Debug.Print " Plotstyle - " & layerKey.PlotStyleName
    Debug.Print " Plottable - " & layerKey.Plottable
    Debug.Print " Removable - " & layerKey.Removeable
Next

End Sub
Sub Example_Linetypes()
    ' This example finds the linetypes collection and
    ' lists all the available linetypes in the collection.

    Dim linetypeColl As AcadLineTypes
    Dim entry As AcadLineType
    Dim msg As String

    ' Return the linetype collection object of the active document
    Set linetypeColl = ThisDrawing.Linetypes
    ' List all available linetypes
    For Each entry In linetypeColl
        msg = msg & entry.name & vbCrLf
    Next
    MsgBox "The linetypes available in this drawing are:" & vbCrLf & msg, vbInformation, "Linetypes Example"
End Sub
Sub Example_LinetypeScale()
    ' This example creates a line and finds the linetype scale
    ' for the line. It then changes the linetype scale, and finally
    ' resets the linetype scale back to the original value.
    Dim startPoint(0 To 2) As Double
    Dim endPoint(0 To 2) As Double
    Dim lineObj As AcadLine
    Dim currLTScale As Double

    ' Create a Line object in model space
    startPoint(0) = 2#: startPoint(1) = 2#: startPoint(2) = 0#
    endPoint(0) = 4#: endPoint(1) = 4#: endPoint(2) = 0#
    Set lineObj = ThisDrawing.ModelSpace.AddLine(startPoint, endPoint)
    lineObj.Update
    currLTScale = lineObj.LinetypeScale
    MsgBox "The linetype scale for the line is:" & lineObj.LinetypeScale, vbInformation, "Linetypes Example"

    ' Set the linetype scale of a Line to .5
    lineObj.LinetypeScale = 0.5
    lineObj.Update
    MsgBox "The new linetype scale for the line is:" & lineObj.LinetypeScale, vbInformation, "Linetypes Example"

    ' Reset the linetype scale of a Line to what is was before
    lineObj.LinetypeScale = currLTScale
    lineObj.Update
    MsgBox "The linetype scale for the line is reset to:" & lineObj.LinetypeScale, vbInformation, "Linetypes Example"
End Sub
**LineWeight Example**

Sub Example_LineWeight()

'This example shows the lineweight of the layer generated by the key

    Dim doc As AecArchBaseDocument
    Dim dbPref As AecArchBaseDatabasePreferences
    Dim cLayerKeyStyles As AecLayerKeyStyles
    Dim layerKeyStyle As AecLayerKeyStyle
    Dim cLayerKeys As AecLayerKeys
    Dim layerKey As AecLayerKey

    Set doc = AecArchBaseApplication.ActiveDocument
    Set cLayerKeyStyles = doc.LayerKeyStyles
    Set dbPref = doc.Preferences
    ' Sets the layer key style to the current layer standard
    Set layerKeyStyle = cLayerKeyStyles.Item(dbPref.LayerStandard)
    Set cLayerKeys = layerKeyStyle.Keys

    For Each layerKey In cLayerKeys
        Debug.Print layerKey.Name
        Debug.Print " Color   - " & layerKey.Color
        Debug.Print " Layer    - " & layerKey.Layer
        Debug.Print " Linetype - " & layerKey.Linetype
        Debug.Print " Lineweight - " & layerKey.Lineweight
        Debug.Print " Plotstyle - " & layerKey.PlotStyleName
        Debug.Print " Plottable - " & layerKey.Plottable
        Debug.Print " Removable - " & layerKey.Removeable
    Next

End Sub
Sub Example_LineweightDisplay()

' This example reads and modifies the preference value which controls
' whether symbol names may include extended character sets, or more
' than 31 characters.
'
' When finished, this example resets the preference value back to
' it's original value.

Dim ACADPref As AcadDatabasePreferences
Dim originalValue As Variant, newValue As Variant

' Get the user preferences object
Set ACADPref = ThisDrawing.preferences

' Read and display the original value
originalValue = ACADPref.LineWeightDisplay
MsgBox "The LineweightDisplay preference is set to: " & originalValue

' Modify the LineweightDisplay preference by toggling the value
ACADPref.LineWeightDisplay = Not (ACADPref.LineWeightDisplay)
newValue = ACADPref.LineWeightDisplay
MsgBox "The LineweightDisplay preference has been set to: " & newValue

' Reset the preference back to it's original value

' * Note: Comment out this last section to leave the change to
' this preference in effect
ACADPref.LineWeightDisplay = originalValue
MsgBox "The LineweightDisplay preference was reset back to: " & originalValue
End Sub
Function Example_Location

This example displays the location of an AEC Entity

Dim obj As Object
Dim pt As Variant
Dim geo As AecGeo

ThisDrawing.Utility.GetEntity obj, pt, "Select an AEC Entity"

If TypeOf obj Is AecGeo Then
    Set geo = obj
    MsgBox "AEC Entity Location: " & geo.Location(0) & ", " & geo.Location(1) & ", " & geo.Location(2), vbInformation, "Location Example"
Else
    MsgBox "Not an AEC Entity", vbExclamation, "Location Example"
End If

End Sub
Sub Example_LowerExtension()

' This example shows the distance of the lower cut plane below building elevation line position

    Dim object As Object
    Dim clip As AecClipVol
    Dim count As Integer

    ' initialize
    count = 0

    For Each object In ThisDrawing.ModelSpace
        If TypeOf object Is AecClipVol Then
            count = count + 1
            Set clip = object
            MsgBox "ClipVol " & count & " Lower Extension is: " & clip.LowerExtension,
        End If
    Next

    If count = 0 Then
        MsgBox "No ClipVol Present in Drawing", vbInformation, "LowerExtension Example"
    End If

End Sub
Sub Example_MaskBlockStyles()

    ' This example will display the number of Mask Block Styles in the current dra

    Dim doc As AecBaseDocument
    Dim app As New AecBaseApplication
    Dim cMaskBlocks As AecMaskBlockStyles

    app.Init ThisDrawing.Application
    Set doc = app.ActiveDocument
    Set cMaskBlocks = doc.MaskBlockStyles
    MsgBox "Number of MaskBlock Styles in this drawing is: " _
    & cMaskBlocks.Count, vbInformation, "MaskBlockStyles Example"

End Sub
**MassGroup Example**

**Examples:**

<table>
<thead>
<tr>
<th>MassGroup Example AecMassElement</th>
</tr>
</thead>
<tbody>
<tr>
<td>MassGroup Example AecMassGroup</td>
</tr>
</tbody>
</table>

```vba
Sub Example_MassGroup_AecMassElement()

    'This example shows the name of the mass' group, if applicable
    Dim object As Object
    Dim mass As AecMassElement
    Dim count As Integer

    'initialize
    count = 0

    For Each object In ThisDrawing.ModelSpace

        If TypeOf object Is AecMassElement Then
            count = count + 1
            Set mass = object
            MsgBox "Mass Element " & count & " Mass Group Name is: " & mass.M
        End If

    Next

    If count = 0 Then
        MsgBox "No Mass Elements Present in Drawing", vbInformation, "MassGroup Example"
    End If

End Sub
```
Sub Example_MassGroup_AecMassGroup()

' This example shows the Mass Group the Mass Element is attached to.

Dim ent As AcadEntity
Dim pt As Variant
Dim mass As AecMassElement
Dim massGroup As AecMassGroup

ThisDrawing.Utility.GetEntity ent, pt, "Select AEC Mass Element"

If TypeOf ent Is AecMassElement Then
    Set mass = ent
    Set massGroup = mass.massGroup
    If Not massGroup Is Nothing Then
        MsgBox "Mass Group is: " & massGroup.Name, vbInformation, "Mass Group Example"
    Else
        MsgBox "Mass Element is not part of a Mass Group", vbInformation, "Mass Group Example"
    End If
Else
    MsgBox "Not an AecMassElement", vbExclamation, "Mass Group Example"
End If

End Sub
MassGroups Example

Sub Example_MassGroups()

    ' This example will display the number of Mass Groups in the current drawing.

    Dim app As New AecBaseApplication
    Dim Aecdoc As AecBaseDocument
    Dim massgroups As AecMassGroups

    app.Init ThisDrawing.Application

    Set Aecdoc = app.ActiveDocument
    Set Aecdoc = AecArchBaseApplication.ActiveDocument
    Set massgroups = Aecdoc.massgroups
    MsgBox "There are " & massgroups.count & " mass groups in this drawing"

    End Sub
MaxActiveViewports Example

Sub Example_MaxActiveViewports()
    ' This example returns the current setting of
    ' MaxActiveViewports. It then changes the value, and finally
    ' it resets the value back to the original setting.

    Dim currMaxActiveViewports As Integer
    Dim newMaxActiveViewports As Integer

    ' Retrieve the current MaxActiveViewports value
    currMaxActiveViewports = ThisDrawing.preferences.MaxActiveViewports
    MsgBox "The current value for MaxActiveViewports is " & currMaxActiveViewports, vbInformation, "MaxActiveViewports Example"

    ' Change the value for MaxActiveViewports
    newMaxActiveViewports = 20
    ThisDrawing.preferences.MaxActiveViewports = newMaxActiveViewports
    MsgBox "The new value for MaxActiveViewports is " & newMaxActiveViewports, vbInformation, "MaxActiveViewports Example"

    ' Reset MaxActiveViewports to its original value
    ThisDrawing.preferences.MaxActiveViewports = currMaxActiveViewports
    MsgBox "The MaxActiveViewports value is reset to " & currMaxActiveViewports, vbInformation, "MaxActiveViewports Example"
End Sub
Sub Example_MeasurementUnit()

' This example returns the MeasurementUnit setting for the current drawing.
Dim dbPref As AecArchBaseDatabasePreferences
Set dbPref = AecArchBaseApplication.ActiveDocument.preferences

' Convert the measurement unit to a string.
Dim unit As String
If dbPref.MeasurementUnit = acEnglish Then
    unit = "English"
Else
    unit = "Metric"
End If

MsgBox "The current value for MeasurementUnit is " & unit, _
    vbInformation, "MeasurementUnit Example"
End Sub
Sub Example_Mirror()
    ' This example creates a light weight polyline
    ' and then mirrors that polyline.

    ' Create the polyline
    Dim plineObj As AcadLWPolyline
    Dim points(0 To 11) As Double
    points(0) = 1: points(1) = 1
    points(2) = 1: points(3) = 2
    points(4) = 2: points(5) = 2
    points(6) = 3: points(7) = 2
    points(8) = 4: points(9) = 4
    points(10) = 4: points(11) = 1
    Set plineObj = ThisDrawing.ModelSpace.AddLightWeightPolyline(points)
    plineObj.Closed = True
    ZoomAll

    ' Define the mirror axis
    Dim point1(0 To 2) As Double
    Dim point2(0 To 2) As Double
    point1(0) = 0: point1(1) = 4.25: point1(2) = 0
    point2(0) = 4: point2(1) = 4.25: point2(2) = 0

    MsgBox "Mirror the polyline.", , "Mirror Example"

    ' Mirror the polyline
    Dim mirrorObj As AcadLWPolyline
    Set mirrorObj = plineObj.Mirror(point1, point2)
    mirrorObj.Color = acRed
    ZoomAll
    MsgBox "Mirror completed.", , "Mirror Example"
End Sub
Sub Example_Mirror3D()
    ' This example creates a box in model space.
    ' It then mirrors the box about a plane and colors
    ' the new box red.

    Dim boxObj As Acad3DSolid
    Dim length As Double, width As Double, height As Double
    Dim center(0 To 2) As Double

    ' Define the box
    center(0) = 5#: center(1) = 5#: center(2) = 0
    length = 5#: width = 7: height = 10#

    ' Create the box (3DSolid) object in model space
    Set boxObj = ThisDrawing.ModelSpace.AddBox(center, length, width, height)

    ' Define the mirroring plane with three points
    Dim mirrorPt1(0 To 2) As Double
    Dim mirrorPt2(0 To 2) As Double
    Dim mirrorPt3(0 To 2) As Double

    mirrorPt1(0) = 1.25: mirrorPt1(1) = 0: mirrorPt1(2) = 0
    mirrorPt2(0) = 1.25: mirrorPt2(1) = 2: mirrorPt2(2) = 0
    mirrorPt3(0) = 1.25: mirrorPt3(1) = 2: mirrorPt3(2) = 2

    ' Mirror the box
    Dim mirrorBoxObj As Acad3DSolid
    Set mirrorBoxObj = boxObj.Mirror3D(mirrorPt1, mirrorPt2, mirrorPt3)
    mirrorBoxObj.Color = acRed

    ' Change the viewing direction of the viewport
    Dim NewDirection(0 To 2) As Double
    NewDirection(0) = -1: NewDirection(1) = -1: NewDirection(2) = 1
    ThisDrawing.ActiveViewport.direction = NewDirection
    ThisDrawing.ActiveViewport = ThisDrawing.ActiveViewport
    ZoomAll
End Sub
ModelSpace Example

Sub Example_ModelSpace()
    ' This example adds a line and a circle to model space.
    ' The line is added using a user-defined variable representing
    ' the model space. The circle is added without using the
    ' user-defined variable. Either use of the ModelSpace
    ' property is valid.

    ' Define the line
    Dim lineObj As AcadLine
    Dim startPoint(0 To 2) As Double
    Dim endPoint(0 To 2) As Double
    startPoint(0) = 0: startPoint(1) = 0: startPoint(2) = 0
    endPoint(0) = 4: endPoint(1) = 4: endPoint(2) = 0

    ' Add the line to model space using the mspace variable
    Dim mspace As AcadModelSpace
    Set mspace = ThisDrawing.ModelSpace
    Set lineObj = mspace.AddLine(startPoint, endPoint)

    ' Define a circle
    Dim circleObj As AcadCircle
    Dim center(0 To 2) As Double
    Dim radius As Double
    center(0) = 4: center(1) = 4: center(2) = 0
    radius = 1

    ' Add the circle to modelspace without using the mspace variable
    Set circleObj = ThisDrawing.ModelSpace.AddCircle(center, radius)

    ZoomAll

End Sub
Modified Example

' This code is placed in a module and is run to initialized the event handler.

Public eh As New EventHandler

Sub Notify()

    Dim obj As AcadObject
    For Each obj In ThisDrawing.ModelSpace
        If TypeOf obj Is AecMassElement Then
            'This sets an event handler (defined below) on the Mass Element
            Set eh.obj = obj
            Exit For
        End If
    Next

End Sub

' This is the Event Handler code
' It is placed in a "Class Module"

Public WithEvents obj As AcadObject

Private Sub obj_Modified(ByVal pObject As IAcadObject)

    Dim ent As AecMassElement
    Set ent = pObject

    Dim loc As Variant
    loc = ent.Location

    Dim str As String
    str = "AecMassElement at (" & loc(0) & ", " & _
          loc(1) & ", " & _
          loc(2) & ")"
MsgBox str, "Object Event"

End Sub
Sub Example_Move()
    ' This example creates a circle and then performs
    ' a move on that circle.

    ' Create the circle
    Dim circleObj As AcadCircle
    Dim center(0 To 2) As Double
    Dim radius As Double
    center(0) = 2#: center(1) = 2#: center(2) = 0#
    radius = 0.5
    Set circleObj = ThisDrawing.ModelSpace.AddCircle(center, radius)
    ZoomAll

    ' Define the points that make up the move vector
    Dim point1(0 To 2) As Double
    Dim point2(0 To 2) As Double
    point1(0) = 0: point1(1) = 0: point1(2) = 0
    point2(0) = 2: point2(1) = 0: point2(2) = 0

    MsgBox "Move the circle 2 units in the X direction.", , "Move Example"

    ' Move the circle
    circleObj.Move point1, point2

    ZoomAll
    MsgBox "Move completed.", , "Move Example"

End Sub
MVBlockRef Example

Examples:

- AecViewBlock
- AecViewBlocks

Sub Example_Name_AecViewBlock()

'This example shows the MVBlockRef's number of colors

    Dim obj As Object
    Dim pt As Variant
    Dim blockRef As AecMVBlockRef

    ThisDrawing.Utility.GetEntity obj, pt, "Select a Multiview Block"
    If TypeOf obj Is AecMVBlockRef Then
        Set blockRef = obj

        MsgBox "Colors of View Blocks MVBlockRef: " & blockRef.viewBlocks.Item(0).MVBlockRef.Color, vbInformation, "MVBlockRef Example"
    Else
        MsgBox "Not a Multiview Block", vbInformation, "MVBlockRef Example"
    End If

End Sub

Sub Example_Name_AecViewBlocks()

'This example shows the number of colors of the owner of the Viewblock collecti

    Dim obj As Object
    Dim pt As Variant
    Dim blockRef As AecMVBlockRef

Dim viewBlocks As AecViewBlocks

ThisDrawing.Utility.GetEntity obj, pt, "Select a Multiview Block"
If TypeOf obj Is AecMVBlockRef Then
  Set blockRef = obj
  Set viewBlocks = blockRef.viewBlocks
  MsgBox "Colors of MVBlockRef Owner: " & viewBlocks.MVBlockRef.Color, vbInformation, "MVBlockRef Example"
Else
  MsgBox "Not a Multiview Block", vbInformation, "MVBlockRef Example"
End If

End Sub
Sub Example_MVBlockStyles()

' This example will display the number of MultiView Block Styles in the current drawing.

Dim doc As AecArchBaseDocument
Dim cMvBlockStyles As AecMVBlockStyles

Set doc = AecArchBaseApplication.ActiveDocument

Set cMvBlockStyles = doc.MVBlockStyles
MsgBox "Number of MVBlock Styles in this drawing is: " & cMvBlockStyles.

End Sub
Name Example

Examples:

1 AecMassGroup
1 AecCamera
1 AecLayerKey
1 AecViewBlock

Sub Example_Name_AecMassGroup()

'This example shows the name of the mass group

    Dim object As Object
    Dim group As AecMassGroup
    Dim count As Integer

    'initialize
    count = 0

    For Each object In ThisDrawing.ModelSpace
        If TypeOf object Is AecMassGroup Then
            count = count + 1
            Set group = object
            MsgBox "Mass Group " & count & " Name is: " & group.Name, vbInformation, "Name Example"
        End If

    Next

    If count = 0 Then
        MsgBox "No Mass Element Groups Present in Drawing", vbInformation, "Name Example"
    End If
Sub Example_Name_AecCamera()

' This example displays the name of a selected camera object

Dim obj As Object
Dim pt As Variant
Dim camera As AecCamera

ThisDrawing.Utility.GetEntity obj, pt, "Select a Camera"

If TypeOf obj Is AecCamera Then
    Set camera = obj
    MsgBox "Name is: " & camera.Name, vbInformation, "Name Example"
Else
    MsgBox "Not a Camera", vbExclamation, "Name Example"
End If

End Sub

Sub Example_Name_AecLayerKey()

'This example shows the name of the layer generated by the key

Dim doc As AecArchBaseDocument
Dim dbPref As AecArchBaseDatabasePreferences
Dim cLayerKeyStyles As AecLayerKeyStyles
Dim layerKeyStyle As AecLayerKeyStyle
Dim cLayerKeys As AecLayerKeys
Dim layerKey As AecLayerKey

Set doc = AecArchBaseApplication.ActiveDocument
Set cLayerKeyStyles = doc.LayerKeyStyles
Set dbPref = doc.Preferences
' Sets the layer key style to the current layer standard
Set layerKeyStyle = cLayerKeyStyles.Item(dbPref.LayerStandard)
Set cLayerKeys = layerKeyStyle.Keys

For Each layerKey In cLayerKeys
    Debug.Print layerKey.Name
    Debug.Print " Color - " & layerKey.Color
    Debug.Print " Layer - " & layerKey.Layer
    Debug.Print " Linetype - " & layerKey.Linetype
    Debug.Print " Lineweight - " & layerKey.Lineweight
    Debug.Print " Plotstyle - " & layerKey.PlotStyleName
    Debug.Print " Plottable - " & layerKey.Plottable
    Debug.Print " Removable - " & layerKey.Removeable
Next

End Sub

Sub Example_Name_AecViewBlock()

    'This example shows the name of the first viewblock of a multiviewblock

    Dim obj As Object
    Dim pt As Variant
    Dim blockRef As AecMVBlockRef
    Dim viewBlocks As AecViewBlocks

    ThisDrawing.Utility.GetEntity obj, pt, "Select a Multiview Block"
    If TypeOf obj Is AecMVBlockRef Then
        Set blockRef = obj
        Set viewBlocks = blockRef.viewBlocks
        MsgBox "Name of View Block 1: " & viewBlocks.Item(0).Name, vbInformation
    Else
        MsgBox "Not a Multiview Block", vbInformation, "Name Example"
    End If

End Sub
Node Example

Sub Example_Node()

    'This example will add anchor a new mass element to a 2D layout grid in the ' drawing.

    Dim grid As AecLayoutGrid2D
    Dim mass As AecMassElement
    Dim pt As Variant
    Dim obj As AcadObject

    ThisDrawing.Utility.GetEntity obj, pt, "Select grid to attach to"
    If TypeOf obj Is AecLayoutGrid2D Then
        Set grid = obj
        Set mass = ThisDrawing.ModelSpace.AddCustomObject("AecMassElement"
        Dim anchor As New AecAnchorEntToLayoutNode
        anchor.Reference = grid
        ' anchor the mass element to the last node on the grid
        Dim lastNode As Long
        lastNode = grid.XNodes.Count * grid.YNodes.Count
        anchor.Node = lastNode
        mass.AttachAnchor anchor
    Else
        MsgBox "No Layout Grid selected", vbInformation, "Node Example"
    End If

End Sub
Sub Example_Nodes()

'This example displays the number of nodes on the layout curve

Dim obj As Object
Dim pt As Variant
Dim layoutCurve As AecLayoutCurve

ThisDrawing.Utility.GetEntity obj, pt, "Select a Node on an AEC Layout Curve"

If TypeOf obj Is AecLayoutCurve Then
    Set layoutCurve = obj
    MsgBox "Number of nodes is: " & layoutCurve.nodes.Count, vbInformation,
Else
    MsgBox "Not a AEC Layout Curve", vbExclamation, "Nodes Example"
End If

End Sub
Normal Example

Sub Example_Normal ()

' This example displays the normal vector of an AEC Entity

Dim obj As Object
Dim pt As Variant
Dim geo As AecGeo

ThisDrawing.Utility.GetEntity obj, pt, "Select an AEC Entity"

If TypeOf obj Is AecGeo Then
    Set geo = obj
    MsgBox "AEC Entity Normal: " & geo.Normal(0) &", " & geo.Normal(1) & vbCrLf & "Normal Example"
Else
    MsgBox "Not an AEC Entity", vbExclamation, "Normal Example"
End If

End Sub
**NorthRotation Example**

Sub Example_NorthRotation()

  ' This example returns the NorthRotation setting for the current drawing.

  Dim dbPref As AecArchBaseDatabasePreferences

  Set dbPref = AecArchBaseApplication.ActiveDocument.Preferences

  MsgBox "The current value for NorthRotation is: " & dbPref.NorthRotation, _
        vbInformation, "NorthRotation Example"

End Sub
ObjectID Example

Sub Example_ObjectID()
    ' This example creates several objects in model space.
    ' It then iterates through model space and displays the
    ' ObjectID for each object found.
    
    ' Create a Ray object in model space
    Dim rayObj As AcadRay
    Dim basePoint(0 To 2) As Double
    Dim SecondPoint(0 To 2) As Double
    basePoint(0) = 3#: basePoint(1) = 3#: basePoint(2) = 0#
    SecondPoint(0) = 1#: SecondPoint(1) = 3#: SecondPoint(2) = 0#
    Set rayObj = ThisDrawing.ModelSpace.AddRay(basePoint, SecondPoint)
    
    ' Create a polyline object in model space
    Dim plineObj As AcadLWPolyline
    Dim points(0 To 5) As Double
    points(0) = 3: points(1) = 7
    points(2) = 9: points(3) = 2
    points(4) = 3: points(5) = 5
    Set plineObj = ThisDrawing.ModelSpace.AddLightWeightPolyline(points)
    plineObj.Closed = True
    
    ' Create a line object in model space
    Dim lineObj As AcadLine
    Dim startPoint(0 To 2) As Double
    Dim endPoint(0 To 2) As Double
    startPoint(0) = 0: startPoint(1) = 0: startPoint(2) = 0
    endPoint(0) = 2: endPoint(1) = 2: endPoint(2) = 0
    Set lineObj = ThisDrawing.ModelSpace.AddLine(startPoint, endPoint)
    
    ' Create a circle object in model space
    Dim circObj As AcadCircle
    Dim centerPt(0 To 2) As Double
    Dim radius As Double
    centerPt(0) = 5: centerPt(1) = 3: centerPt(2) = 0
    radius = 3
Set circObj = ThisDrawing.ModelSpace.AddCircle(centerPt, radius)

' Create an ellipse object in model space
Dim ellObj As AcadEllipse
Dim majAxis(0 To 2) As Double
Dim center(0 To 2) As Double
Dim radRatio As Double
center(0) = 5#: center(1) = 5#: center(2) = 0#
majAxis(0) = 10: majAxis(1) = 20#: majAxis(2) = 0#
radRatio = 0.3
Set ellObj = ThisDrawing.ModelSpace.AddEllipse(center, majAxis, radRatio)

ZoomAll

' Iterate through the model space collection and display
' the ObjectID of each entity found.

Dim entObjectID As Long
Dim entry As AcadEntity
For Each entry In ThisDrawing.ModelSpace
    entObjectID = entry.objectID
    entry.Highlight (True)
    MsgBox "The ObjectID of this object is " & entObjectID, vbInformation, "C
    entry.Highlight (False)
Next
End Sub
ObjectIDToObject Example

Sub Example_ObjectIDToObject()
    ' This example creates a spline in model space. It then returns the
    ' handle for the spline. The spline is returned from the handle, and
    ' then colored.

    ' Create the spline
    Dim splineObj As AcadSpline
    Dim startTan(0 To 2) As Double
    Dim endTan(0 To 2) As Double
    Dim fitPoints(0 To 8) As Double

    startTan(0) = 0.5: startTan(1) = 0.5: startTan(2) = 0
    endTan(0) = 0.5: endTan(1) = 0.5: endTan(2) = 0
    fitPoints(0) = 1: fitPoints(1) = 1: fitPoints(2) = 0
    fitPoints(3) = 5: fitPoints(4) = 5: fitPoints(5) = 0
    fitPoints(6) = 10: fitPoints(7) = 0: fitPoints(8) = 0
    Set splineObj = ThisDrawing.ModelSpace.AddSpline(fitPoints, startTan, endTan)
    ZoomAll

    ' Find the objectID of the spline
    Dim objectID As Long
    objectID = splineObj.objectID
    MsgBox "The objectID of the Spline is: " & splineObj.objectID, , "ObjectIDToObject Example"

    ' Find an object from a given objectID
    Dim tempObj As AcadObject
    Set tempObj = ThisDrawing.ObjectIdToObject(objectID)

    ' Now use the newly initialized object variable to color the object red
    tempObj.Color = acRed
    ThisDrawing.Regen True
    MsgBox "The Spline is now red.", , "ObjectIDToObject Example"

End Sub
Sub Example_ObjectName()
  ' This example creates several objects in model space.
  ' It then iterates through model space and displays the
  ' Object Name for each object found.

  ' Create a Ray object in model space
  Dim rayObj As AcadRay
  Dim basePoint(0 To 2) As Double
  Dim SecondPoint(0 To 2) As Double
  basePoint(0) = 3#: basePoint(1) = 3#: basePoint(2) = 0#
  SecondPoint(0) = 1#: SecondPoint(1) = 3#: SecondPoint(2) = 0#
  Set rayObj = ThisDrawing.ModelSpace.AddRay(basePoint, SecondPoint)

  ' Create a polyline object in model space
  Dim plineObj As AcadLWPolyline
  Dim points(0 To 5) As Double
  points(0) = 3: points(1) = 7
  points(2) = 9: points(3) = 2
  points(4) = 3: points(5) = 5
  Set plineObj = ThisDrawing.ModelSpace.AddLightWeightPolyline(points)
  plineObj.Closed = True

  ' Create a line object in model space
  Dim lineObj As AcadLine
  Dim startPoint(0 To 2) As Double
  Dim endPoint(0 To 2) As Double
  startPoint(0) = 0: startPoint(1) = 0: startPoint(2) = 0
  endPoint(0) = 2: endPoint(1) = 2: endPoint(2) = 0
  Set lineObj = ThisDrawing.ModelSpace.AddLine(startPoint, endPoint)

  ' Create a circle object in model space
  Dim circObj As AcadCircle
  Dim centerPt(0 To 2) As Double
  Dim radius As Double
  centerPt(0) = 20: centerPt(1) = 30: centerPt(2) = 0
  radius = 3
Set circObj = ThisDrawing.ModelSpace.AddCircle(centerPt, radius)

' Create an ellipse object in model space
Dim ellObj As AcadEllipse
Dim majAxis(0 To 2) As Double
Dim center(0 To 2) As Double
Dim radRatio As Double
center(0) = 5#: center(1) = 5#: center(2) = 0#
majAxis(0) = 10: majAxis(1) = 20#: majAxis(2) = 0#
radRatio = 0.3
Set ellObj = ThisDrawing.ModelSpace.AddEllipse(center, majAxis, radRatio)

ZoomAll

' Iterate through the model space collection and display
' the object name of each entity found.

Dim objName As String
Dim entry As AcadEntity
For Each entry In ThisDrawing.ModelSpace
    objName = entry.ObjectName
    entry.Highlight (True)
    MsgBox "The name of this object is " & objName, vbInformation, "ObjectName Example"
    entry.Highlight (False)
Next
End Sub
Sub Example_ObjectSortByPlotting()
    ' This example reads and modifies the preference value which controls
    ' the sorting of drawing objects by plotting order.
    ' When finished, this example resets the preference value back to
    ' it's original value.

    Dim ACADPref As AcadDatabasePreferences
    Dim originalValue As Variant, newValue As Variant

    ' Get the user preferences object
    Set ACADPref = ThisDrawing.preferences

    ' Read and display the original value
    originalValue = ACADPref.ObjectSortByPlotting
    MsgBox "The ObjectSortByPlotting preference is set to: " & originalValue

    ' Modify the ObjectSortByPlotting preference by toggling the value
    ACADPref.ObjectSortByPlotting = Not (ACADPref.ObjectSortByPlotting)
    newValue = ACADPref.ObjectSortByPlotting
    MsgBox "The ObjectSortByPlotting preference has been set to: " & newValue

    ' Reset the preference back to it's original value
    
    ' * Note: Comment out this last section to leave the change to
    ' this preference in effect
    ACADPref.ObjectSortByPlotting = originalValue
    MsgBox "The ObjectSortByPlotting preference was reset back to: " & originalValue
End Sub
Sub Example_ObjectSortByPSOutput()
' This example reads and modifies the preference value which controls
' the sorting of drawing objects by PostScript output order.
' When finished, this example resets the preference value back to
' it's original value.

Dim ACADPref As AcadDatabasePreferences
Dim originalValue As Variant, newValue As Variant

' Get the user preferences object
Set ACADPref = ThisDrawing.preferences

' Read and display the original value
originalValue = ACADPref.ObjectSortByPSOutput
MsgBox "The ObjectSortByPSOutput preference is set to: " & originalValue

' Modify the ObjectSortByPSOutput preference by toggling the value
ACADPref.ObjectSortByPSOutput = Not (ACADPref.ObjectSortByPSOutput)
newValue = ACADPref.ObjectSortByPSOutput
MsgBox "The ObjectSortByPSOutput preference has been set to: " & newValue

' Reset the preference back to it's original value
'
' * Note: Comment out this last section to leave the change to
' this preference in effect
ACADPref.ObjectSortByPSOutput = originalValue
MsgBox "The ObjectSortByPSOutput preference was reset back to: " & originalValue
End Sub
Sub Example_ObjectSortByRedraws()
    ' This example reads and modifies the preference value which controls
    ' the sorting of drawing objects by redraw order.
    ' When finished, this example resets the preference value back to
    ' it's original value.

    Dim ACADPref As AcadDatabasePreferences
    Dim originalValue As Variant, newValue As Variant

    ' Get the user preferences object
    Set ACADPref = ThisDrawing.preferences

    ' Read and display the original value
    originalValue = ACADPref.ObjectSortByRedraws
    MsgBox "The ObjectSortByRedraws preference is set to: " & originalValue

    ' Modify the ObjectSortByRedraws preference by toggling the value
    ACADPref.ObjectSortByRedraws = Not (ACADPref.ObjectSortByRedraws)
    newValue = ACADPref.ObjectSortByRedraws
    MsgBox "The ObjectSortByRedraws preference has been set to: " & newValue

    ' Reset the preference back to it's original value
    
    ' * Note: Comment out this last section to leave the change to
    '       this preference in effect
    ACADPref.ObjectSortByRedraws = originalValue
    MsgBox "The ObjectSortByRedraws preference was reset back to: " & originalValue
End Sub
Sub Example_ObjectSortByRegens()
    ' This example reads and modifies the preference value which controls
    ' the sorting of drawing objects by regeneration order.
    ' When finished, this example resets the preference value back to
    ' it's original value.

    Dim ACADPref As AcadDatabasePreferences
    Dim originalValue As Variant, newValue As Variant

    ' Get the user preferences object
    Set ACADPref = ThisDrawing.preferences

    ' Read and display the original value
    originalValue = ACADPref.ObjectSortByRegens
    MsgBox "The ObjectSortByRegens preference is set to: " & originalValue

    ' Modify the ObjectSortByRegens preference by toggling the value
    ACADPref.ObjectSortByRegens = Not (ACADPref.ObjectSortByRegens)
    newValue = ACADPref.ObjectSortByRegens
    MsgBox "The ObjectSortByRegens preference has been set to: " & newValue

    ' Reset the preference back to it's original value
    ' *
    ' * Note: Comment out this last section to leave the change to
    ' this preference in effect
    ACADPref.ObjectSortByRegens = originalValue
    MsgBox "The ObjectSortByRegens preference was reset back to: " & originalValue
End Sub
Sub Example_ObjectSortBySelection()
  ' This example reads and modifies the preference value which controls
  ' the sorting of drawing objects by object selection.
  ' When finished, this example resets the preference value back to
  ' it's original value.

  Dim ACADPref As AcadDatabasePreferences
  Dim originalValue As Variant, newValue As Variant

  ' Get the user preferences object
  Set ACADPref = ThisDrawing.preferences

  ' Read and display the original value
  originalValue = ACADPref.ObjectSortBySelection
  MsgBox "The ObjectSortBySelection preference is set to: " & originalValue

  ' Modify the ObjectSortBySelection preference by toggling the value
  ACADPref.ObjectSortBySelection = Not (ACADPref.ObjectSortBySelection)
  newValue = ACADPref.ObjectSortBySelection
  MsgBox "The ObjectSortBySelection preference has been set to: " & newValue

  ' Reset the preference back to it's original value
  ' *
  ' * Note: Comment out this last section to leave the change to
  ' this preference in effect
  ACADPref.ObjectSortBySelection = originalValue
  MsgBox "The ObjectSortBySelection preference was reset back to: " & originalValue
End Sub
Sub Example_ObjectSortBySnap()
    ' This example reads and modifies the preference value which controls
    ' the sorting of drawing objects by object snap.
    ' When finished, this example resets the preference value back to
    ' it's original value.

    Dim ACADPref As AcadDatabasePreferences
    Dim originalValue As Variant, newValue As Variant

    ' Get the user preferences object
    Set ACADPref = ThisDrawing.preferences

    ' Read and display the original value
    originalValue = ACADPref.ObjectSortBySnap
    MsgBox "The ObjectSortBySnap preference is set to: " & originalValue

    ' Modify the ObjectSortBySnap preference by toggling the value
    ACADPref.ObjectSortBySnap = Not (ACADPref.ObjectSortBySnap)
    newValue = ACADPref.ObjectSortBySnap
    MsgBox "The ObjectSortBySnap preference has been set to: " & newValue

    ' Reset the preference back to it's original value

    ' * Note: Comment out this last section to leave the change to
    ' this preference in effect
    ACADPref.ObjectSortBySnap = originalValue
    MsgBox "The ObjectSortBySnap preference was reset back to: " & originalValue
End Sub
Sub Example_Offset()

'This example will find the offset of the MVBlock tag to the end of the leader.

Dim obj As AcadObject
Dim bubble As AecMVBlockRef
Dim anchor As AecAnchor
Dim leaderAnchor As AecAnchorLeadEntToNode
Dim pt As Variant

ThisDrawing.Utility.GetEntity obj, pt, "Select bubble"
If TypeOf obj Is AecMVBlockRef Then
    Set bubble = obj
    Set anchor = obj.GetAnchor
    If TypeOf anchor Is AecAnchorLeadEntToNode Then
        Set leaderAnchor = anchor
        Dim offset As Variant
        Dim offsetString As String

        offset = leaderAnchor.offset
        offsetString = offset(0) & ", " & offset(1) & ", " & offset(2)
        MsgBox "Offset from node = " & offsetString, vbInformation, "Offset Example"
    Else
        MsgBox "Not anchored to column grid", vbInformation, "Offset Example"
    End If
Else
    MsgBox "Not a bubble", vbInformation, "Offset Example"
End If

End Sub
OLELaunch Example

Sub Example_OLELaunch()
' This example reads and modifies the preference value which controls
' whether to launch the parent application when plotting OLE objects.
' When finished, this example resets the preference value back to
' it's original value.

Dim ACADPref As AcadDatabasePreferences
Dim originalValue As Variant, newValue As Variant

' Get the user preferences object
Set ACADPref = ThisDrawing.preferences

' Read and display the original value
originalValue = ACADPref.OLELaunch
MsgBox "The OLELaunch preference is set to: " & originalValue

' Modify the OLELaunch preference by toggling the value
ACADPref.OLELaunch = Not (ACADPref.OLELaunch)
newValue = ACADPref.OLELaunch
MsgBox "The OLELaunch preference has been set to: " & newValue

' Reset the preference back to it's original value
'
' * Note: Comment out this last section to leave the change to
' this preference in effect
ACADPref.OLELaunch = originalValue
MsgBox "The OLELaunch preference was reset back to: " & originalValue
End Sub
Operation Example

Examples:

1 AecMassElement
1 AecMassGroup

Sub Example_Operation_AecMassElement()

'This example shows the operation of a mass element?

Dim object As Object
Dim mass As AecMassElement
Dim count As Integer

'initialize
count = 0

For Each object In ThisDrawing.ModelSpace

    If TypeOf object Is AecMassElement Then
        count = count + 1
        Set mass = object

        Select Case mass.Operation
            Case aecMassOperationAdd
                MsgBox "Mass Element " & count & " Operation: Add", vbInformation
            Case aecMassOperationIntersect
                MsgBox "Mass Element " & count & " Operation: Intersect", vbInformation
            Case aecMassOperationSubtract
                MsgBox "Mass Element " & count & " Operation: Subtract", vbInformation
        End Select
    End If

End If

Next
If count = 0 Then
    MsgBox "No Mass Elements Present in Drawing", vbInformation, "Operatic
End If

End Sub

Sub Example_Operation_AecMassGroup()

'This example shows the operation of a mass group?

Dim object As Object
Dim group As AecMassGroup
Dim count As Integer

'initialize
count = 0

For Each object In ThisDrawing.ModelSpace

    If TypeOf object Is AecMassGroup Then
        count = count + 1
        Set group = object

        Select Case group.Operation
            Case aecMassOperationAdd
                MsgBox "Mass Group " & count & " Operation: Add", vbInformation,
            Case aecMassOperationIntersect
                MsgBox "Mass Group " & count & " Operation: Intersect", vbInformation,
            Case aecMassOperationSubtract
                MsgBox "Mass Group " & count & " Operation: Subtract", vbInformation
        End Select

    End If

Next
If count = 0 Then
    MsgBox "No Mass Element Groups Present in Drawing", vbInformation, "C
End If

End Sub
OverridesEnabled Example

Sub Example OverridesEnabled()
    Dim db As New AecBaseDatabase
    Dim cLayerKeyStyles As AecLayerKeyStyles
    Dim layerKeyStyle As aeclayerkeystyle
    MsgBox "Layer Overrides Enabled: " & layerKeyStyle.OverridesEnabled, vbInformation
End Sub
OverrideSettings Example

Sub Example_OverrideSettings()

    Dim db As New AecBaseDatabase
    Dim cLayerKeyStyles As AecLayerKeyStyles
    Dim layerKeyStyle As aeclayerkeystyle
    Dim overrideSettings As AecLayerOverrideSettings
    db.Init ThisDrawing.Database

    Set cLayerKeyStyles = db.LayerKeyStyles
    Set layerKeyStyle = cLayerKeyStyles.Item(0)
    Set overrideSettings = layerKeyStyle.overrideSettings

End Sub
 OWNERID Example

Sub Example_OWNERID()
  ' This example creates an MText object in model space
  ' and then finds the OwnerID for the object.

  Dim MTextObj As AcadMText
  Dim corner(0 To 2) As Double
  Dim width As Double
  Dim text As String
  corner(0) = 0#: corner(1) = 10#: corner(2) = 0#
  width = 10
  text = "This is the text String for the mtext Object"

  ' Creates the mtext Object
  Set MTextObj = ThisDrawing.ModelSpace.AddMText(corner, width, text)
  ZoomAll

  MsgBox "The OwnerID for the MText object is: " & MTextObj.OwnerID

End Sub
Sub Example_PaperSpace()
    ' This example sets the variable pspace to the paperspace entity collection.
    Dim pspace As AcadPaperSpace
    Set pspace = ThisDrawing.PaperSpace

End Sub
Sub Example_PlotConfigurations()
    ' This example will access the PlotConfigurations collection for the current drawing,
    ' add a plot configuration, and list basic information about the plot configurations in the drawing.

    Dim PlotConfigurations As AcadPlotConfigurations
    Dim PlotConfiguration As AcadPlotConfiguration
    Dim msg As String

    ' Get PlotConfigurations collection from document object
    Set PlotConfigurations = ThisDrawing.PlotConfigurations

    ' If there aren't any plot configurations, then we'll add one
    If PlotConfigurations.count = 0 Then
        '*** Customize the new configuration to your satisfaction ***
        PlotConfigurations.Add "NEW_CONFIGURATION"
    End If

    msg = vbCrLf & vbCrLf ' Start with a space

    ' Get the names of the plot configurations in this drawing
    For Each PlotConfiguration In PlotConfigurations
        msg = msg & PlotConfiguration.name & vbCrLf
    Next

    ' Display a list of available plot configurations
    MsgBox "There are " & PlotConfigurations.count & " plot configuration(s) in " & ThisDrawing.WindowTitle & ":" & msg
End Sub
Sub Example_PlotStyleName()

'This example shows the plotstyle name of the layer generated by the key

    Dim doc As AecArchBaseDocument
    Dim dbPref As AecArchBaseDatabasePreferences
    Dim cLayerKeyStyles As AecLayerKeyStyles
    Dim layerKeyStyle As AecLayerKeyStyle
    Dim cLayerKeys As AecLayerKeys
    Dim layerKey As AecLayerKey

    Set doc = AecArchBaseApplication.ActiveDocument
    Set cLayerKeyStyles = doc.LayerKeyStyles
    Set dbPref = doc.Preferences
    ' Sets the layer key style to the current layer standard
    Set layerKeyStyle = cLayerKeyStyles.Item(dbPref.LayerStandard)
    Set cLayerKeys = layerKeyStyle.Keys

    For Each layerKey In cLayerKeys
        Debug.Print layerKey.Name
        Debug.Print " Color   - " & layerKey.Color
        Debug.Print " Layer   - " & layerKey.Layer
        Debug.Print " Linetype - " & layerKey.Linetype
        Debug.Print " Lineweight - " & layerKey.Lineweight
        Debug.Print " Plotstyle - " & layerKey.PlotStyleName
        Debug.Print " Plottable - " & layerKey.Plottable
        Debug.Print " Removable - " & layerKey.Removeable
    Next

End Sub
'Plottable Example

Sub Example_Plottable()

'This example shows whether the layer is plottable

    Dim doc As AecArchBaseDocument
    Dim dbPref As AecArchBaseDatabasePreferences
    Dim cLayerKeyStyles As AecLayerKeyStyles
    Dim layerKeyStyle As AecLayerKeyStyle
    Dim cLayerKeys As AecLayerKeys
    Dim layerKey As AecLayerKey

    Set doc = AecArchBaseApplication.ActiveDocument
    Set cLayerKeyStyles = doc.LayerKeyStyles
    Set dbPref = doc.Preferences
    ' Sets the layer key style to the current layer standard
    Set layerKeyStyle = cLayerKeyStyles.Item(dbPref.LayerStandard)
    Set cLayerKeys = layerKeyStyle.Keys

    For Each layerKey In cLayerKeys
        Debug.Print layerKey.Name
        Debug.Print "Color    - " & layerKey.Color
        Debug.Print "Layer    - " & layerKey.Layer
        Debug.Print "LineType  - " & layerKey.Linetype
        Debug.Print "Lineweight - " & layerKey.Lineweight
        Debug.Print "Plotstyle - " & layerKey.PlotStyleName
        Debug.Print "Plottable - " & layerKey.Plottable
        Debug.Print "Removable - " & layerKey.Removeable
    Next

End Sub
Sub Example_PolygonStyles()

' This example will display the number of Polygon Styles in the current drawing.

Dim doc As AecArchBaseDocument
Dim cPolygonStyles As AecPolygonStyles

Set doc = AecArchBaseApplication.ActiveDocument
Set cPolygonStyles = doc.PolygonStyles

MsgBox "Number of Polygon Styles in this drawing is: " & cPolygonStyles.Count,
End Sub
Sub Example_Preferences()

    ' This example returns the ElevationPrecision setting for the current drawing.
    Dim dbPref As AecDatabasePreferences
    Set dbPref = AecArchBaseApplication.ActiveDocument.Preferences

    MsgBox "The current value for ElevationPrecision is: " & dbPref.ElevationPrecision
    vbInformation, "ElevationPrecision Example"

End Sub
Profile Example

Examples:

1. **AecProfileStyle**
2. **AecRing**
3. **AecRings**

---

**Sub** Example_Profile_AecProfileStyle()

' This example will display the area of the first profile style in the collection

*Dim* db As New AecBaseDatabase  
*Dim* pStyle As AecProfileStyle  

db.Init ThisDrawing.Database

*Set* pStyle = db.ProfileStyles.Item(0)  
MsgBox "Profile Area: " & pStyle.Profile.Area, vbInformation, "Profile Example"

**End Sub**

---

**Sub** Example_Profile_AecRing()

' This example will display the area of the ring's owner profile

*Dim* db As New AecBaseDatabase  
*Dim* ring As AecRing  

db.Init ThisDrawing.Database

*Set* ring = db.ProfileStyles.Item(0).Profile.rings.Item(0)  
MsgBox "Ring Profile Area: " & ring.Profile.Area, vbInformation, "Profile Example"
Sub Example_Profile_AecRings()

' This example will display the area of the profile that owns the ring collection

Dim db As New AecBaseDatabase
Dim pStyle As AecProfileStyle

db.Init ThisDrawing.Database

Set pStyle = db.ProfileStyles.Item(0)
MsgBox "Rings Profile Area: " & pStyle.Profile.rings.Profile.Area, vbInformation

End Sub
Sub Example_ProfileStyle()

    Dim ent As Object
    Dim pt As Variant
    Dim mass As AecMassElement

    ThisDrawing.Utility.GetEntity ent, pt, "Select AEC Mass Element"

    If TypeOf ent Is AecMassElement Then
        Set mass = ent
        MsgBox "ProfileStyle is: " & mass.ProfileStyle.Name, vbInformation, "ProfileStyle Example"
    Else
        MsgBox "Not an AecMassElement", vbExclamation, "ProfileStyle Example"
    End If

End Sub
Sub Example_ProfileStyleName ()

' This example shows the profile style name for the mass element

    Dim object As Object
    Dim mass As AecMassElement
    Dim count As Integer

    ' initialize
    count = 0

    For Each object In ThisDrawing.ModelSpace
        If TypeOf object Is AecMassElement Then
            count = count + 1
            Set mass = object
            MsgBox "Mass Element " & count & " Name is: " & mass.ProfileStyleName,
            End If
        Next

    If count = 0 Then
        MsgBox "No Mass Elements Present in Drawing", vbInformation, "Profile Style Name Example"
    End If

End Sub
ProfileStyles Example

Sub Example_ProfileStyles()

' This example will display the number of Profile Styles in the current drawing.

Dim doc As AecArchBaseDocument
Dim cProfileStyles As AecProfileStyles

Set doc = AecArchBaseApplication.ActiveDocument

Set cProfileStyles = doc.ProfileStyles
MsgBox "Number of Profile Styles in this drawing is: " & cProfileStyles.Count,
End Sub
Sub Example_ProjectName()

' This example displays the ProjectName setting for the current drawing

Dim dbPref As AecArchBaseDatabasePreferences

Set dbPref = AecArchBaseApplication.ActiveDocument.preferences

MsgBox "The setting for ProjectName is: " & dbPref.ProjectName, _
        vbInformation, "ProjectName Example"

End Sub
Sub Example_Radius()

    'This example shows the radius of a mass element

    Dim mass As AecMassElement
    Dim pt1(0 To 2) As Double
    pt1(0) = 100: pt1(1) = 100: pt1(2) = 0

    Set mass = ThisDrawing.ModelSpace.AddCustomObject("AecMassElement")
    mass.Type = aec MassElementTypeCone
    mass.Height = 100
    mass.Radius = 50
    mass.Location = pt1

    MsgBox "Mass Element Radius = " & mass.Radius, vbInformation, "Radius E:

End Sub
Sub Example_Reference()

    ' This example will make a mass element and anchor it to a line.

    Dim mass As AecMassElement
    Dim line As AcadLine
    Dim anchor As New AecAnchorEntToCurve
    Dim pt1(0 To 2) As Double
    Dim pt2(0 To 2) As Double

    pt2(0) = 1000: pt2(1) = 1000

    ' this makes a new mass element 120 x 120 x 120 at 0,0,0
    Set mass = ThisDrawing.ModelSpace.AddCustomObject("AecMassElement")
    ' draws a line from 0,0 to 1000,1000
    Set line = ThisDrawing.ModelSpace.AddLine(pt1, pt2)

    anchor.Reference = line ' set the line as the object (curve) to anchor to
    mass.AttachAnchor anchor ' attach the anchor to the mass element

End Sub
Sub Example_RegisteredApplications()
    ' This example finds the current RegisteredApplications collection and
    ' adds a new RegisteredApplication to that collection.
    Dim RegAppColl As AcadRegisteredApplications
    Set RegAppColl = ThisDrawing.RegisteredApplications

    ' Create a RegisteredApp named "TEST" in the current drawing
    Dim RegAppObj As AcadRegisteredApplication
    Set RegAppObj = RegAppColl.Add("TEST")
    MsgBox "A new registered application called " & RegAppObj.name & " has been added.
End Sub
Sub Example_ReleaseAnchor()

' This example will release the anchor attached to an AEC Entity.

Dim ent As AcadEntity
Dim geo As AecGeo
Dim pt As Variant
Dim anchor As AecAnchor

ThisDrawing.Utility.GetEntity ent, pt, "Selected anchored object"
If TypeOf ent Is AecGeo Then
    Set geo = ent
    Set anchor = geo.GetAnchor
    If anchor Is Nothing Then
        MsgBox "AEC Entity is not anchored", vbInformation, "ReleaseAnchor Example"
    Else
        geo.ReleaseAnchor ' releases the anchor from the entity
        MsgBox "AEC Entity is released", vbInformation, "ReleaseAnchor Example"
    End If
Else
    MsgBox "AEC Entity not selected", vbInformation, "ReleaseAnchor Example"
End If

End Sub
Sub Example_Remove_LayerKeyStyle()

' This example adds a new layer key style to the layer key styles collection, then removes the new style.

Dim app As New AecBaseApplication
Dim doc As AecBaseDocument
Dim layerkeystyles As AecLayerKeyStyles
Dim layerkeystyle As AecLayerKeyStyle

Dim msg As String

' Initialize.
app.Init ThisDrawing.Application
Set doc = app.ActiveDocument

' Get the layer key styles collection and note the number of objects in the collection.
Set layerkeystyles = doc.layerkeystyles
msg = "There were " & layerkeystyles.Count _
  & " layer key styles in this drawing." & vbCrLf

' Add a new layer key style to the collection.
Set layerkeystyle = layerkeystyles.Add("ExampleStyle")
layerkeystyle.Description = "This is a test"

msg = msg & "After adding a layer key style, there are " _
  & layerkeystyles.Count & " of them."
MsgBox msg, vbInformation, "Remove Example"

' Note the number of objects in the layer key styles collection now.
msg = "The new layer key style had " & layerkeystyle.Keys.Count & " keys--"

' Remove the layer key style that was just added.
layerkeystyles.Remove ("ExampleStyle")
msg = msg & " but I've since removed the style..."
MsgBox msg, vbInformation, "Remove Example"

End Sub
Sub Example_Removeable()

' This example shows whether the layer is removeable

    Dim doc As AecArchBaseDocument
    Dim dbPref As AecArchBaseDatabasePreferences
    Dim cLayerKeyStyles As AecLayerKeyStyles
    Dim layerKeyStyle As AecLayerKeyStyle
    Dim cLayerKeys As AecLayerKeys
    Dim layerKey As AecLayerKey

    Set doc = AecArchBaseApplication.ActiveDocument
    Set cLayerKeyStyles = doc.LayerKeyStyles
    Set dbPref = doc.Preferences
    ' Sets the layer key style to the current layer standard
    Set layerKeyStyle = cLayerKeyStyles.Item(dbPref.LayerStandard)
    Set cLayerKeys = layerKeyStyle.Keys

    For Each layerKey In cLayerKeys
        Debug.Print layerKey.Name
        Debug.Print " Color   - " & layerKey.Color
        Debug.Print " Layer   - " & layerKey.Layer
        Debug.Print " Linetype - " & layerKey.Linetype
        Debug.Print " Lineweight - " & layerKey.Lineweight
        Debug.Print " Plotstyle - " & layerKey.PlotStyleName
        Debug.Print " Plottable - " & layerKey.Plottable
        Debug.Print " Removable - " & layerKey.Removeable
        Next

End Sub
Sub Example_RenderSmoothness()
    ' This example returns the current setting of
    ' RenderSmoothness. It then changes the value, and finally
    ' it resets the value back to the original setting.
    Dim currRenderSmoothness As Double
    Dim newRenderSmoothness As Double

    ' Retrieve the current RenderSmoothness value
    currRenderSmoothness = ThisDrawing.preferences.RenderSmoothness
    MsgBox "The current value for RenderSmoothness is " & currRenderSmoothness

    ' Change the value for RenderSmoothness
    newRenderSmoothness = 2.5
    ThisDrawing.preferences.RenderSmoothness = newRenderSmoothness
    MsgBox "The new value for RenderSmoothness is " & newRenderSmoothness

    ' Reset RenderSmoothness to its original value
    ThisDrawing.preferences.RenderSmoothness = currRenderSmoothness
    MsgBox "The RenderSmoothness value is reset to " & currRenderSmoothness
End Sub
**ResizeOffset Example**

Sub Example_ResizeOffset()

' This example will take a mass element anchored to a 2D Layout Grid, and
' apply a Resize Offset to it.

Dim obj As AcadObject
Dim anchor As AecAnchorEntToLayoutCell
Dim pt As Variant
ThisDrawing.Utility.GetEntity obj, pt, "Select anchored element"
If TypeOf obj Is AecMassElement Then
    Dim mass As AecMassElement
    Set mass = obj
    If TypeOf mass.GetAnchor Is AecAnchorEntToLayoutCell Then
        Set anchor = mass.GetAnchor
        anchor.ResizeOffset = -22
        anchor.ApplyResize = False
    End If
End If

End Sub
Sub Example_RightOffset()

    ' This example returns the right offset of the selected object
    ' to the grid assembly.

    ' Use this example with a drawing that contains a window
    ' assembly and one or more AEC objects attached to the
    ' assembly.

    Dim ent As AcadEntity
    Dim geo As AecGeo
    Dim anchor As AecAnchor

    Dim offset As String

    On Error Resume Next    ' Handle errors in code.

    ' Prompt user to select an object.
    ThisDrawing.Utility.GetEntity ent, pt, "Select object anchored to window assembly:

    ' Make sure user selected an AEC object, and that the object
    ' is anchored to a grid assembly.
    If ent Is Nothing Then
        MsgBox "Nothing was selected.", vbExclamation, "RightOffset Example"
    ElseIf TypeOf ent Is AecGeo Then
        Set geo = ent

    ' Get the anchor the object is attached to.
    Set anchor = geo.GetAnchor
    On Error GoTo 0
    If anchor Is Nothing Then
        MsgBox "Selected object is not anchored.", vbExclamation, "RightOffset Example"
    ElseIf Not TypeOf anchor Is AecAnchorEntToGridAssembly Then
        MsgBox "Object is anchored, but not to a grid assembly.", vbExclamation
    Else
        MsgBox "Right offset of object: " & anchor.RightOffset, vbInformation, "RightOffset Example"
End If
Else
    MsgBox "Object selected is not an AEC entity.", vbInformation, "RightOffset"
End If

End Sub
Rings Example

Sub Example_Rings()

' This example shows the number of rings of the profile of an AecPolygon

    Dim obj As Object
    Dim pt As Variant
    Dim poly As AecPolygon

    ThisDrawing.Utility.GetEntity obj, pt, "Select an AECPolygon"
    If TypeOf obj Is AecPolygon Then
        Set poly = obj
        MsgBox "Profile Rings: " & poly.Profile.rings.Count, vbInformation, "Rings Example"
    Else
        MsgBox "Not a Polygon or no Profile Found", vbInformation, "Rings Example"
    End If

End Sub
Sub Example_Rise()

' This example shows the rise of the mass element if gable shape

Dim object As Object
Dim mass As AecMassElement
Dim count As Integer

' Initialize
count = 0

For Each object In ThisDrawing.ModelSpace
    If TypeOf object Is AecMassElement Then
        count = count + 1
        Set mass = object
        MsgBox "Mass Element " & count & " Rise is: " & mass.Rise, vbInformation, "Rise Example"
    End If
Next

If count = 0 Then
    MsgBox "No Mass Elements Present in Drawing", vbInformation, "Rise Example"
End If

End Sub
Sub Example_Rotate()
  ' This example creates a light weight polyline
  ' and then rotates that polyline.

  ' Create the polyline
  Dim plineObj As AcadLWPolyline
  Dim points(0 To 11) As Double
  points(0) = 1: points(1) = 2
  points(2) = 1: points(3) = 3
  points(4) = 2: points(5) = 3
  points(6) = 3: points(7) = 3
  points(8) = 4: points(9) = 4
  points(10) = 4: points(11) = 2
  Set plineObj = ThisDrawing.ModelSpace.AddLightWeightPolyline(points)
  plineObj.Closed = True
  ZoomAll

  MsgBox "Rotate the polyline by 45 degrees.", , "Rotate Example"

  ' Define the rotation
  Dim basePoint(0 To 2) As Double
  Dim rotationAngle As Double
  basePoint(0) = 4: basePoint(1) = 4.25: basePoint(2) = 0
  rotationAngle = 0.7853981 ' 45 degrees

  ' Rotate the polyline
  plineObj.Rotate basePoint, rotationAngle

  ZoomAll
  MsgBox "Rotation completed.", , "Rotate Example"
End Sub
Rotate3D Example

Sub Example_Rotate3D()
' This example creates a box in model space.
' It then rotates the box about an axis.

Dim boxObj As Acad3DSolid
Dim length As Double, width As Double, height As Double
Dim center(0 To 2) As Double

' Define the box
center(0) = 5#: center(1) = 5#: center(2) = 0
length = 5#: width = 7: height = 10#

' Create the box (3DSolid) object in model space
Set boxObj = ThisDrawing.ModelSpace.AddBox(center, length, width, height)

' Change the viewing direction of the viewport
Dim NewDirection(0 To 2) As Double
NewDirection(0) = -1: NewDirection(1) = -1: NewDirection(2) = 1
ThisDrawing.ActiveViewport.direction = NewDirection
ThisDrawing.ActiveViewport = ThisDrawing.ActiveViewport
ThisDrawing.Regen True

' Define the rotation axis with two points
Dim rotatePt1(0 To 2) As Double
Dim rotatePt2(0 To 2) As Double
Dim rotateAngle As Double

rotatePt1(0) = -3: rotatePt1(1) = 4: rotatePt1(2) = 0
rotatePt2(0) = -3: rotatePt2(1) = -4: rotatePt2(2) = 0
rotateAngle = 30
rotateAngle = rotateAngle * 3.141592 / 180#

' Draw a line between the two axis points so that it is visible.
' This is optional. It is not required for the rotation.
Dim axisLine As AcadLine
Set axisLine = ThisDrawing.ModelSpace.AddLine(rotatePt1, rotatePt2)
axisLine.Update
MsgBox "Rotate the box 30 degrees about the axis shown.", "Rotate3D Example"

' Rotate the box
boxObj.Rotate3D rotatePt1, rotatePt2, rotateAngle
ThisDrawing.Regen True
MsgBox "The box is rotated 30 degrees.", "Rotate3D Example"

End Sub
Rotation Example

Sub Example_Rotation()

'This example shows the rotation reference angle for an AEC Entity

    Dim object As Object
    Dim pt As Variant
    Dim geo As AecGeo

    ThisDrawing.Utility.GetEntity object, pt, "Select AEC Entity:"

    If TypeOf object Is AecGeo Then
        Set geo = object
        Dim rotString As String
        rotString = ThisDrawing.Utility.AngleToString(geo.Rotation, acDegrees, 4)
        MsgBox "AEC Entity Rotation is: " & rotString, vbInformation, "Rotation Example"
    Else
        MsgBox "No AEC Entity selected.", vbInformation, "Rotation Example"
    End If

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 AecArchBaseDatabasePreferences
Set dbPref = AecArchBaseApplication.ActiveDocument.preferences

dbPref.TextHeight = 0.25
dbPref.SaveAsDefault

MsgBox "The new default value for TextHeight is: " & dbPref.TextHeight, _
    vbInformation, "SaveAsDefault Example"

End Sub
Sub Example_ScaleEntity()
' This example creates a lightweight polyline
' and then scales that polyline.

' Create the polyline
Dim plineObj As AcadLWPolyline
Dim points(0 To 11) As Double
points(0) = 1: points(1) = 2
points(2) = 1: points(3) = 3
points(4) = 2: points(5) = 3
points(6) = 3: points(7) = 3
points(8) = 4: points(9) = 4
points(10) = 4: points(11) = 2
Set plineObj = ThisDrawing.ModelSpace.AddLightWeightPolyline(points)
plineObj.Closed = True
ZoomAll

MsgBox "Scale the polyline by 0.5", , "ScaleEntity Example"

' Define the scale
Dim basePoint(0 To 2) As Double
Dim scaleFactor As Double
basePoint(0) = 4: basePoint(1) = 4.25: basePoint(2) = 0
scaleFactor = 0.5

' Scale the polyline
plineObj.ScaleEntity basePoint, scaleFactor

ZoomAll
MsgBox "Scale completed.", , "ScaleEntity Example"

End Sub
Sub Example_ScaleOnInsert()

' This example returns the ScaleOnInsert setting for the current drawing.
Dim dbPref As AecArchBaseDatabasePreferences
Set dbPref = AecArchBaseApplication.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
**ScaleX Example**

**Examples:**

1. `AecMaskBlockRef`
2. `AecMVBlockRef`

---

**Sub Example_ScaleX_AecMaskBlockRef()**

```
' This example shows the X scale of the mask block

Dim obj As Object
Dim pt As Variant
Dim mask As AecMaskBlockRef

ThisDrawing.Utility.GetEntity obj, pt, "Select Mask Block"

If TypeOf obj Is AecMaskBlockRef Then
    Set mask = obj
    MsgBox "Scale X is: " & mask.ScaleX, vbInformation, "ScaleX Example"
Else
    MsgBox "Not a Mask Block", vbExclamation, "ScaleX Example"
End If

End Sub
```

---

**Sub Example_ScaleX_AecMVBlockRef()**

```
Dim ent As AcadEntity
Dim pt As Variant
Dim mvBlock As AecMVBlockRef

ThisDrawing.Utility.GetEntity ent, pt, "Select AEC Multi-View Block"
```
If TypeOf ent Is AecMVBlockRef Then
    Set mvBlock = ent
    MsgBox "Scale X is: " & mvBlock.ScaleX, vbInformation, "ScaleX Example"
Else
    MsgBox "Not an AecMVBlockRef", vbExclamation, "ScaleX Example"
End If

End Sub
Sub Example_ScaleXY ()

    Dim ring As AecRing
    Dim profile As AecProfile

    Dim doc As AecArchBaseDocument
    Set doc = AecArchBaseApplication.ActiveDocument
    Dim cprofiles As AecProfileStyles
    Dim profileStyle As AecProfileStyle

    Set cprofiles = doc.ProfileStyles
    Set profileStyle = cprofiles.Item(0)

    Set profileStyle.profile = profile.ScaleXY(2, 1)

End Sub
**ScaleY Example**

**Examples:**

1. `AecMaskBlockRef`

2. `AecMVBlockRef`

---

**Sub** `Example_ScaleY_AecMaskBlockRef()`

```
' This example shows the Y scale of the mask block

Dim obj As Object
Dim pt As Variant
Dim mask As AecMaskBlockRef

ThisDrawing.Utility.GetEntity obj, pt, "Select Mask Block"

If TypeOf obj Is AecMaskBlockRef Then
    Set mask = obj
    MsgBox "Scale Y is: " & mask.ScaleY, vbInformation, "ScaleY Example"
Else
    MsgBox "Not a Mask Block", vbExclamation, "ScaleY Example"
End If
```

End Sub

---

**Sub** `Example_ScaleY_AecMVBlockRef()`

```
Dim ent As AcadEntity
Dim pt As Variant
Dim mvBlock As AecMVBlockRef

ThisDrawing.Utility.GetEntity ent, pt, "Select AEC Multi-View Block"
```
If TypeOf ent Is AecMVBlockRef Then
    Set mvBlock = ent
    MsgBox "Scale Y is: " & mvBlock.ScaleY, vbInformation, "ScaleY Example"
Else
    MsgBox "Not an AecMVBlockRef", vbExclamation, "ScaleY Example"
End If

End Sub
ScaleZ Example

Examples:

1 AecMaskBlockRef
1 AecMVBlockRef

Sub Example_ScaleZ_AecMaskBlockRef()

' This example shows the Z scale of the mask block

    Dim obj As Object
    Dim pt As Variant
    Dim mask As AecMaskBlockRef

    ThisDrawing.Utility.GetEntity obj, pt, "Select Mask Block"

    If TypeOf obj Is AecMaskBlockRef Then
        Set mask = obj
        MsgBox "Scale Z is: " & mask.ScaleZ, vbInformation, "ScaleZ Example"
    Else
        MsgBox "Not a Mask Block", vbExclamation, "ScaleZ Example"
    End If

End Sub

Sub Example_ScaleZ_AecMVBlockRef()

    Dim ent As AcadEntity
    Dim pt As Variant
    Dim mvBlock As AecMVBlockRef

    ThisDrawing.Utility.GetEntity ent, pt, "Select AEC Multi-View Block"
If TypeOf ent Is AecMVBlockRef Then
    Set mvBlock = ent
    MsgBox "Scale Z is: " & mvBlock.ScaleZ, vbInformation, "ScaleZ Example"
Else
    MsgBox "Not an AecMVBlockRef", vbExclamation, "ScaleZ Example"
End If

End Sub
Sub Example_SegmentPerPolyline()
    ' This example returns the current setting of
    ' SegmentPerPolyline. It then changes the value, and finally
    ' it resets the value back to the original setting.

    Dim currSegmentPerPolyline As Integer
    Dim newSegmentPerPolyline As Integer

    ' Retrieve the current SegmentPerPolyline value
    currSegmentPerPolyline = ThisDrawing.preferences.SegmentPerPolyline
    MsgBox "The current value for SegmentPerPolyline is " & currSegmentPerPolyline, vbInformation, "SegmentPerPolyline Example"

    ' Change the value for SegmentPerPolyline
    newSegmentPerPolyline = 2001
    ThisDrawing.preferences.SegmentPerPolyline = newSegmentPerPolyline
    MsgBox "The new value for SegmentPerPolyline is " & newSegmentPerPolyline, vbInformation, "SegmentPerPolyline Example"

    ' Reset SegmentPerPolyline to its original value
    ThisDrawing.preferences.SegmentPerPolyline = currSegmentPerPolyline
    MsgBox "The SegmentPerPolyline value is reset to " & currSegmentPerPolyline, vbInformation, "SegmentPerPolyline Example"
End Sub
Sub Example_SelfIntersects()

' This example shows if an AecPolygon self intersects

    Dim obj As Object
    Dim pt As Variant
    Dim poly As AecPolygon

    ThisDrawing.Utility.GetEntity obj, pt, "Select an AECPolygon"
    If TypeOf obj Is AecPolygon Then
        Set poly = obj
        MsgBox "Profile SelfIntersects?: " & poly.Profile.SelfIntersects, vbInformation
    Else
        MsgBox "Not a Polygon or no Profile Found", vbInformation
    End If

End Sub
Sub Example_SetXdata()
  ' This example creates a line and attaches extended data to that line.
  
  ' Create the line
  Dim lineObj As AcadLine
  Dim startPt(0 To 2) As Double, endPt(0 To 2) As Double
  startPt(0) = 1#: startPt(1) = 1#: startPt(2) = 0#
  endPt(0) = 5#: endPt(1) = 5#: endPt(2) = 0#
  Set lineObj = ThisDrawing.ModelSpace.AddLine(startPt, endPt)
  ZoomAll
  
  ' Initialize all the xdata values. Note that first data in the list should be
  ' application name and first datatype code should be 1001
  Dim DataType(0 To 9) As Integer
  Dim Data(0 To 9) As Variant
  Dim reals3(0 To 2) As Double
  Dim worldPos(0 To 2) As Double
  
  DataType(0) = 1001: Data(0) = "Test_Application"
  DataType(1) = 1000: Data(1) = "This is a test for xdata"
  
  DataType(2) = 1003: Data(2) = "0" ' layer
  DataType(3) = 1040: Data(3) = 1.23479137438413E+40 ' real
  DataType(4) = 1041: Data(4) = 1237324938 ' distance
  DataType(5) = 1070: Data(5) = 32767 ' 16 bit Integer
  DataType(6) = 1071: Data(6) = 32767 ' 32 bit Integer
  DataType(7) = 1042: Data(7) = 10 ' scaleFactor
  
  reals3(0) = -2.95: reals3(1) = 100: reals3(2) = -20
  DataType(8) = 1010: Data(8) = reals3 ' real
  
  worldPos(0) = 4: worldPos(1) = 400.99999999: worldPos(2) = 2.798989
  DataType(9) = 1011: Data(9) = worldPos ' world space position
  
  ' Attach the xdata to the line
  lineObj.SetXData DataType, Data
' Return the xdata for the line
Dim xdataOut As Variant
Dim xtypeOut As Variant
lineObj.GetXData "", xtypeOut, xdataOut

End Sub
Sub Example_Side1()

' This example shows the length of cut plane along the starting edge

    Dim object As Object
    Dim clip As AecClipVol
    Dim count As Integer

    ' Initialize
    count = 0

    For Each object In ThisDrawing.ModelSpace
        If TypeOf object Is AecClipVol Then
            count = count + 1
            Set clip = object
            MsgBox "ClipVol " & count & " Side 1 is: " & clip.Side1, vbInformation,
        End If
    Next

    If count = 0 Then
        MsgBox "No ClipVol Present in Drawing", vbInformation, "Side1 Example"
    End If

End Sub
Sub Example_Side2()

'This example shows the length of cut plane along the ending edge

    Dim object As Object
    Dim clip As AecClipVol
    Dim count As Integer

    'initialize
    count = 0

    For Each object In ThisDrawing.ModelSpace
        If TypeOf object Is AecClipVol Then
            count = count + 1
            Set clip = object
            MsgBox "ClipVol " & count & " Side 2 is: " & clip.Side2, vbInformation,
            End If
        Next

    If count = 0 Then
        MsgBox "No ClipVol Present in Drawing", vbInformation, "Side2 Example'
    End If

End Sub
SolidFill Example

Sub Example_SolidFill()
    ' This example reads and modifies the preference value which controls
    ' whether multilines, traces, solids, all hatches (including solid-fill)
    ' and wide polylines are filled in.
    '
    ' When finished, this example resets the preference value back to
    ' it's original value.
    
    Dim ACADPref As AcadDatabasePreferences
    Dim originalValue As Variant, newValue As Variant
    
    ' Get the user preferences object
    Set ACADPref = ThisDrawing.preferences
    
    ' Read and display the original value
    originalValue = ACADPref.SolidFill
    MsgBox "The SolidFill preference is set to: " & originalValue
    
    ' Modify the SolidFill preference by toggling the value
    ACADPref.SolidFill = Not (ACADPref.SolidFill)
    newValue = ACADPref.SolidFill
    MsgBox "The SolidFill preference has been set to: " & newValue
    
    ' Reset the preference back to it's original value
    
    ' * Note: Comment out this last section to leave the change to
    ' this preference in effect
    ACADPref.SolidFill = originalValue
    MsgBox "The SolidFill preference was reset back to: " & originalValue
End Sub
Sub Example_StartOffset()

'This example displays the start offset of a selected AEC Layout Curve object

Dim obj As Object  
Dim pt As Variant  
Dim layoutCurve As AecLayoutCurve

ThisDrawing.Utility.GetEntity obj, pt, "Select a Node on an AEC Layout Curve'

If TypeOf obj Is AecLayoutCurve Then
    Set layoutCurve = obj
    MsgBox "StartOffset is: " & layoutCurve.StartOffset, vbInformation, "StartOffset Example"
Else
    MsgBox "Not a AEC Layout Curve", vbExclamation, "StartOffset Example"
End If

End Sub
Sub Example_Style()

' This example shows the style of an AEC Entity

Dim obj As Object
Dim pt As Variant
ThisDrawing.Utility.GetEntity obj, pt, "Select an AEC Object"

If TypeOf obj Is AecGeo Then
    On Error Resume Next
    Dim name As String
    name = obj.Style.name
    ' If the entity doesn't use a style, this will fail
    If (name "") Then
        MsgBox "Style is: " & name, vbInformation, "Style Example"
    Else
        MsgBox "AEC Entity does not have a style", vbInformation, "Style Example"
    End If
Else
    MsgBox "Not a AEC Entity", vbExclamation, "Style Example"
End If

End Sub
StyleName Example

Sub Example_StyleName()

' This example shows the style name of an AEC Entity

Dim obj As Object
Dim pt As Variant
ThisDrawing.Utility.GetEntity obj, pt, "Select an AEC Object"

If TypeOf obj Is AecGeo Then
    On Error Resume Next
    Dim name As String
    name = obj.StyleName
    ' If the entity doesn't use a style, this will fail
    If (name "") Then
        MsgBox "Style is: " & name, vbInformation, "Style Example"
    Else
        MsgBox "AEC Entity does not have a style", vbInformation, "Style Example"
    End If
Else
    MsgBox "Not a AEC Entity", vbExclamation, "Style Example"
End If

End Sub
Subtract Example

Sub Example_Subtract ()

' This example adds two profiles from rings, and subtracts the second from the first

On Error Resume Next
Dim pointList1(0 To 9) As Double
Dim pointList2(0 To 7) As Double
pointList1(0) = 0: pointList1(1) = 0
pointList1(2) = 1: pointList1(3) = 0
pointList1(4) = 1: pointList1(5) = 1
pointList1(6) = 0: pointList1(7) = 1
pointList1(8) = 0: pointList1(9) = 0

pointList2(0) = 0.5: pointList2(1) = 0.5
pointList2(2) = 0.067: pointList2(3) = -0.25
pointList2(4) = 0.933: pointList2(5) = -0.25
pointList2(6) = 0.5: pointList2(7) = 0.5

Dim ring1 As AecRing
Dim ring2 As AecRing
Dim profile1 As New AecProfile
Dim profile2 As New AecProfile

Dim doc As AecArchBaseDocument
Set doc = AecArchBaseApplication.ActiveDocument
Dim cprofiles As AecProfileStyles
Dim profileStyle As AecProfileStyle

Set cprofiles = doc.ProfileStyles
Set profileStyle = cprofiles.Item("SubtractRing")
If profileStyle Is Nothing Then
    Set profileStyle = cprofiles.Add("SubtractRing")
End If
Set ring1 = profile1.Rings.Add
Set ring2 = profile2.Rings.Add
ring1.FromPoints (pointList1)
ring2.FromPoints (pointList2)

profile1.Subtract profile2

Set profileStyle.profile = profile1

End Sub
Sub Example_TextFrameDisplay()
    ' This example returns the current setting of TextFrameDisplay. It then changes the value, and finally it resets the value back to the original setting.

    Dim currTextFrameDisplay As Boolean

    ' Retrieve the current TextFrameDisplay value
    currTextFrameDisplay = ThisDrawing.preferences.TextFrameDisplay
    MsgBox "The current value for TextFrameDisplay is " & preferences.TextFrameDisplay

    ' Change the value for TextFrameDisplay
    ThisDrawing.preferences.TextFrameDisplay = Not (currTextFrameDisplay)
    MsgBox "The new value for TextFrameDisplay is " & preferences.TextFrameDisplay

    ' Reset TextFrameDisplay to its original value
    ThisDrawing.preferences.TextFrameDisplay = currTextFrameDisplay
    MsgBox "The TextFrameDisplay value is reset to " & preferences.TextFrameDisplay
End Sub
Sub Example_TextHeight()

' This example displays the TextHeight setting for the current drawing.

Dim dbPref As AecArchBaseDatabasePreferences

Set dbPref = AecArchBaseApplication.ActiveDocument.preferences

MsgBox "The current value for TextHeight is: " & dbPref.TextHeight, _
vbInformation, "TextHeight Example"

End Sub
Sub Example_TextStyles()
    ' This example finds the current Textstyles collection and
    ' adds a new Text style to that collection.

    Dim TextColl As AcadTextStyles
    Set TextColl = ThisDrawing.TextStyles

    ' Create a Text style named "TEST" in the current drawing
    Dim textStyle As AcadTextStyle
    Set textStyle = TextColl.Add("TEST")
    MsgBox "A new Text style called " & textStyle.name & " has been added to the collection.", vbInformation, "Textstyles Example"
End Sub
**TopOffset Example**

Sub Example_TopOffset()

' This example modifies the top offset of the selected object
' in relation to its anchor point on the grid assembly. It
' prompts the user to select the object to be modified,
' and then it prompts the user to specify the number of inches
' to adjust the offset by. The number can be either positive
' or negative.

' Use this example with a drawing that contains a window
' assembly and one or more AEC objects attached to the
' assembly.

Dim ent As AcadEntity
Dim geo As AecGeo
Dim anchor As AecAnchor

Dim offset As String
Dim offset_adjust As Double

On Error Resume Next
   ' Handle errors in code.

' Prompt user to select an object.
ThisDrawing.Utility.GetEntity ent, pt, "Select object anchored to window assem"

' Make sure user selected an AEC object, and that the object
' is anchored to a grid assembly.
If ent Is Nothing Then
   MsgBox "Nothing was selected.", vbExclamation, "TopOffset Example"
ElseIf TypeOf ent Is AecGeo Then
   Set geo = ent

   ' Get the anchor the object is attached to.
   Set anchor = geo.GetAnchor
   On Error GoTo 0
ElseIf TypeOf ent Is Nothing Then
   MsgBox "Nothing was selected.", vbExclamation, "TopOffset Example"

End If
MsgBox "Selected object is not anchored.", vbExclamation, "TopOffset Example"

ElseIf Not TypeOf anchor Is AecAnchorEntToGridAssembly Then
    MsgBox "Object is anchored, but not to a grid assembly.", vbExclamation
Else
    ' AdjustSizing must be set to True in order for offset change to take effect.
    anchor.AdjustSizing = True
    MsgBox "Top offset of object was: " & anchor.TopOffset, vbInformation,

    ' Prompt user to specify amount to adjust offset by.
    offset_adjust = ThisDrawing.Utility.GetReal("Enter the number of inches

    ' Change offset by specified amount.
    anchor.TopOffset = anchor.TopOffset + offset_adjust
    ThisDrawing.Regen (acActiveViewport)
    MsgBox "New top offset is: " & anchor.TopOffset, vbInformation, "TopOffset Example"
End If
Else
    MsgBox "Object selected is not an AEC entity.", vbInformation, "TopOffset Example"
End If

End Sub
Sub Example_TransformBy()
    ' This example creates a line and rotates it 90 degrees
    ' using a transformation matrix.

    ' Create a line
    Dim lineObj As AcadLine
    Dim startPt(0 To 2) As Double
    Dim endPt(0 To 2) As Double
    startPt(0) = 2: startPt(1) = 1: startPt(2) = 0
    endPt(0) = 5: endPt(1) = 1: endPt(2) = 0
    Set lineObj = ThisDrawing.ModelSpace.AddLine(startPt, endPt)
    lineObj.Update

    ' Initialize the transMat variable with a transformation matrix
    ' that will rotate an object by 90 degrees about the point(0,0,0)
    ' (More examples of transformation matrices are listed below)
    Dim transMat(0 To 3, 0 To 3) As Double
    transMat(0, 0) = 0#: transMat(0, 1) = -1#: transMat(0, 2) = 0#: transMat(0, 3) =
    transMat(1, 0) = 1#: transMat(1, 1) = 0#: transMat(1, 2) = 0#: transMat(1, 3) =
    transMat(2, 0) = 0#: transMat(2, 1) = 0#: transMat(2, 2) = 1#: transMat(2, 3) =
    transMat(3, 0) = 0#: transMat(3, 1) = 0#: transMat(3, 2) = 0#: transMat(3, 3) =

    ' Transform the line using the defined transformation matrix
    MsgBox "Transform the line.", , "TransformBy Example"
    lineObj.TransformBy (transMat)
    ZoomAll
    MsgBox "The line is transformed.", , "TransformBy Example"

    ' More examples of transformation matrices:

    ' Rotation Matrix: 90 Degrees about point 0,0,0
    ' 0.000000  -1.000000  0.000000  0.000000
    ' 1.000000   0.000000  0.000000  0.000000
    ' 0.000000   0.000000  1.000000  0.000000
' 0.000000  0.000000  0.000000  1.000000

' Rotation Matrix: 45 Degrees about point 5,5,0
  ' 0.707107  -0.707107  0.000000  5.000000
  ' 0.707107   0.707107  0.000000 -2.071068
  ' 0.000000  0.000000  1.000000  0.000000
  ' 0.000000  0.000000  0.000000  1.000000

' Translation Matrix: move an object by 10,10,0
  ' 1.000000  0.000000  0.000000  10.000000
  ' 0.000000  1.000000  0.000000  10.000000
  ' 0.000000  0.000000  1.000000  0.000000
  ' 0.000000  0.000000  0.000000  1.000000

' Scaling Matrix: scale by 10,10 at point 0,0,0
  ' 10.000000  0.000000  0.000000  0.000000
  ' 0.000000  10.000000  0.000000  0.000000
  ' 0.000000  0.000000  10.000000  0.000000
  ' 0.000000  0.000000  0.000000  1.000000

' Scaling Matrix: scale by 10 at point 2,2
  ' 10.000000  0.000000  0.000000 -18.000000
  ' 0.000000  10.000000  0.000000 -18.000000
  ' 0.000000  0.000000  10.000000  0.000000
  ' 0.000000  0.000000  0.000000  1.000000

End Sub
Type Example

Examples:

1 AecLayoutCurve

1 AecMassElement

Sub Example_Type_AecLayoutCurve()

'This example displays the spacing rule used on a selected aec layout curve obj-

Dim obj As Object
Dim pt As Variant
Dim layoutCurve As AecLayoutCurve
Dim layoutType As AecLayoutType
Dim str As String

ThisDrawing.Utility.GetEntity obj, pt, "Select a Node on an AEC Layout Curve'

If TypeOf obj Is AecLayoutCurve Then
    Set layoutCurve = obj
    layoutType = layoutCurve.Type
    Select Case layoutType
    Case aecLayoutTypeAutoSpacingBay
        str = "Bay Spacing"
    Case aecLayoutTypeAutoSpacingEven
        str = "Even Spacing"
    Case aecLayoutTypeManualSpacing
        str = "Manual Spacing"
    End Select

    MsgBox "Layout rule is: " & str, vbInformation, "Type Example"
Else
    MsgBox "Not a AEC Layout Curve", vbExclamation, "Type Example"
End If
Sub Example_Type_AecMassElement()

' This example shows the type of mass element

Dim object As Object
Dim mass As AecMassElement
Dim count As Integer

' initialize
count = 0

For Each object In ThisDrawing.ModelSpace

    If TypeOf object Is AecMassElement Then
        count = count + 1
        Set mass = object

        Select Case mass.Type
            Case aecMassElementTypeArch
                MsgBox "Mass Element " & count & " Type: Arch", vbInformation, "Type Example"
            Case aecMassElementTypeBarrelVault
                MsgBox "Mass Element " & count & " Type: Barrel Vault", vbInformation, "Type Example"
            Case aecMassElementTypeBox
                MsgBox "Mass Element " & count & " Type: Box", vbInformation, "Type Example"
            Case aecMassElementTypeCone
                MsgBox "Mass Element " & count & " Type: Cone", vbInformation, "Type Example"
            Case aecMassElementTypeCylinder
                MsgBox "Mass Element " & count & " Type: Cylinder", vbInformation, "Type Example"
            Case aecMassElementTypeDome
                MsgBox "Mass Element " & count & " Type: Dome", vbInformation, "Type Example"
            Case aecMassElementTypeDoric
                MsgBox "Mass Element " & count & " Type: Doric", vbInformation, "Type Example"
            Case aecMassElementTypeExtrusion
                MsgBox "Mass Element " & count & " Type: Extrusion", vbInformation, "Type Example"
            Case aecMassElementTypeGable
                MsgBox "Mass Element " & count & " Type: Gable", vbInformation, "Type Example"
        End Select
    End If

Next object

End Sub
Case aecMassElementTypeIsoscelesTriangle
    MsgBox "Mass Element " & count & " Type: Isosceles Triangle", vbInformation
Case aecMassElementTypePyramid
    MsgBox "Mass Element " & count & " Type: Pyramid", vbInformation
Case aecMassElementTypeRevolution
    MsgBox "Mass Element " & count & " Type: Revolution", vbInformation
Case aecMassElementTypeRightTriangle
    MsgBox "Mass Element " & count & " Type: Right Triangle", vbInformation
Case aecMassElementTypeSphere
    MsgBox "Mass Element " & count & " Type: Sphere", vbInformation, ' End Select

End If

Next

If count = 0 Then
    MsgBox "No Mass Elements Present in Drawing", vbInformation, "Type Ex
End If

End Sub
Sub Example_UserCoordinateSystems()
    ' This example finds the current UserCoordinateSystems collection and
    ' adds a new UCS to that collection.

    Dim UCSColl As AcadUCSs
    Set UCSColl = ThisDrawing.UserCoordinateSystems

    ' Create a UCS named "TEST" in the current drawing
    Dim ucsObj As AcadUCS
    Dim origin(0 To 2) As Double
    Dim xAxisPnt(0 To 2) As Double
    Dim yAxisPnt(0 To 2) As Double

    ' Define the UCS
    origin(0) = 4#: origin(1) = 5#: origin(2) = 3#
    xAxisPnt(0) = 5#: xAxisPnt(1) = 5#: xAxisPnt(2) = 3#
    yAxisPnt(0) = 4#: yAxisPnt(1) = 6#: yAxisPnt(2) = 3#

    ' Add the UCS to the UserCoordinatesSystems collection
    Set ucsObj = UCSColl.Add(origin, xAxisPnt, yAxisPnt, "TEST")

    MsgBox "A new UCS called " & ucsObj.name & " has been added to the User Coordinate Systems collection.", vbInformation, "UserCoordinateSystems Example"
End Sub
Sub Example_Update()
    ' This example creates a line in model space. It displays a
    ' MsgBox immediately before the call to update the line and
    ' again after the call.
    Dim lineObj As AcadLine
    Dim startPoint(0 To 2) As Double
    Dim endPoint(0 To 2) As Double

    ' Create a Line object in model space
    startPoint(0) = 2#: startPoint(1) = 2#: startPoint(2) = 0#
    endPoint(0) = 4#: endPoint(1) = 4#: endPoint(2) = 0#
    Set lineObj = ThisDrawing.ModelSpace.AddLine(startPoint, endPoint)
    lineObj.Color = acRed

    MsgBox "Before the update.", , "Update Example"
    ' The following code draws an object in AutoCAD window
    lineObj.Update
    MsgBox "After the update.", , "Update Example"

    ' The following call updates the entire drawing
    ThisDrawing.Application.Update

End Sub
Sub Example_UseModelExtents()

' This example determines if the height and lower extension are set automatically using model extents

    Dim object As Object
    Dim clip As AecClipVol
    Dim count As Integer

    ' initialize
    count = 0

    For Each object In ThisDrawing.ModelSpace
        If TypeOf object Is AecClipVol Then
            count = count + 1
            Set clip = object
            MsgBox "ClipVol " & count & " Use Model Extents?: " & clip.UseModelExtents,
        End If
    Next

    If count = 0 Then
        MsgBox "No ClipVol Present in Drawing", vbInformation, "UseModelExtents Example"
    End If

End Sub
Sub Example_UseNodeCS()

    ' This example attaches a Mass Element to a 2D Layout Grid, and uses the Node coordinate system.

    Dim obj As AcadObject
    Dim pt As Variant
    ThisDrawing.Utility.GetEntity obj, pt, "Select Layout Grid"

    If TypeOf obj Is AecLayoutGrid2D Then
        Dim grid As AecLayoutGrid2D
        Set grid = obj
        Dim mass As AecMassElement
        Set mass = ThisDrawing.ModelSpace.AddCustomObject("AecMassElement"
        Dim anchor As New AecAnchorEntToLayoutNode
        anchor.Reference = grid
        anchor.Node = 1
        anchor.UseNodeCS = True
        mass.AttachAnchor anchor
    Else
        MsgBox "No Layout Grid selected", vbInformation, "UseNodeCS Example"
    End If

End Sub
Valid Example

Examples:

1 AecProfile

1 AecRing

Sub Example_Valid_AecProfile()

' This example shows if an AecPolygon is valid

    Dim obj As Object
    Dim pt As Variant
    Dim poly As AecPolygon

    ThisDrawing.Utility.GetEntity obj, pt, "Select an AECPolygon"
    If TypeOf obj Is AecPolygon Then
        Set poly = obj
        MsgBox "Profile Valid: " & poly.Profile.Valid, vbInformation, "Valid Examp"
        Else
        MsgBox "Not a Polygon or no Profile Found", vbInformation, "Valid Examp"
    End If

End Sub

Sub Example_Valid_AecRing()

' This example will display the validity of the ring

    Dim db As New AecBaseDatabase
    Dim ring As AecRing

    db.Init ThisDrawing.Database
Set ring = db.ProfileStyles.Item(0).Profile.rings.Item(0)
MsgBox "Ring Valid?: " & ring.Valid, vbInformation, "Valid Example"

End Sub
Value Example

Examples:

- **AecLayerOverrideSetting**
- **AecLayoutCurve**

---

**Sub** Example_Value_AecLayerOverrideSetting()

'This example shows the name of the owner of the override setting

```vba
Dim db As New AecBaseDatabase
Dim setting As AecLayerOverrideSetting

db.Init ThisDrawing.Database

Set setting = db.LayerKeyStyles.Item(0).overrideSettings.Item(0)
MsgBox "Setting Value: " & setting.Value, vbInformation, "Value Example"

End Sub
```

---

**Sub** Example_Value_AecLayoutCurve()

'This example shows either the bay spacing or the number of nodes on the layout

```vba
Dim obj As Object
Dim pt As Variant
Dim layoutCurve As AecLayoutCurve
Dim str As String

ThisDrawing.Utility.GetEntity obj, pt, "Select a Node on an AEC Layout Curve"

If TypeOf obj Is AecLayoutCurve Then
    Set layoutCurve = obj
```
Select Case layoutCurve.Type
    Case aecLayoutTypeAutoSpacingBay
        str = "Bay spacing = " & layoutCurve.Value
    Case aecLayoutTypeAutoSpacingEven
        str = "Number of even spaces = " & layoutCurve.Value
    Case aecLayoutTypeManualSpacing
        str = "Manual Spacing"
End Select

MsgBox str, vbInformation, "Value Example"
Else
    MsgBox "Not a AEC Layout Curve", vbExclamation, "Value Example"
End If

End Sub
Sub Example_VerticalScale()

' This example will display the vertical scale of the database.

Dim db As AecArchBaseDatabase
Dim pref As AecArchBaseDatabasePreferences

db.Init ThisDrawing.Database

Set pref = db.Preferences
MsgBox "Vertical Scale of the database: " & pref.VerticalScale, vbInformation

"VerticalScale Example"

End Sub
Sub Example_ViewBlocks()

    Dim ent As AcadEntity
    Dim pt As Variant
    Dim mvBlock As AecMVBlockRef
    Dim cViewBlocks As AecViewBlocks

    ThisDrawing.Utility.GetEntity ent, pt, "Select AEC Multi-View Block"

    If TypeOf ent Is AecMVBlockRef Then
        Set mvBlock = ent
        Set cViewBlocks = mvBlock.viewBlocks
        MsgBox "Number of View blocks is: " & cViewBlocks.Count, vbInformation
    Else
        MsgBox "Not an AecMVBlockRef", vbExclamation, "StyleName Example"
    End If

End Sub
Sub Viewports()
    ' This example finds the current Viewports collection and
    ' adds a new viewport to that collection.

    Dim viewportColl As AcadViewports
    Set viewportColl = ThisDrawing.Viewports

    ' Create a viewport named "TEST" in the current drawing
    Dim viewportObj As AcadViewport
    Set viewportObj = viewportColl.Add("TEST")
    MsgBox "A new viewport called " & viewportObj.name & " has been added to"
End Sub
Views Example

Sub Example_Views()
    ' This example finds the current views collection and
    ' adds a new view to that collection.
    Dim viewColl As AcadViews
    Set viewColl = ThisDrawing.Views

    ' Create a view named "TEST" in the current drawing
    Dim viewObj As AcadView
    Set viewObj = viewColl.Add("TEST")
    MsgBox "A new view called " & viewObj.name & " has been added to the view"
End Sub
Visible Example

Sub Example_Visible_AecBaseObject()

' This example adds an AecPolygon object to the current drawing,
' displays the object, then turns the object's Visible property
' to False and redisplays the drawing.

Dim obj As AcadObject
Dim pt As Variant
Dim polygon As AecPolygon

' Add an AecPolygon object to the drawing.
Set polygon = ThisDrawing.ModelSpace.AddCustomObject("AecPolygon")

' Select a location for the object onscreen.
pt = ThisDrawing.Utility.GetPoint(, "Select the insertion point:"
If Err.Number  0 Then
    MsgBox ("error when getting a point." & vbCrLf)
    Exit Sub
End If

' Place the object at the specified point, then display
' a message so that users can see the object.
polygon.Location = pt
polygon.Update
MsgBox ("AecPolygon added to drawing. Now you see it...")

' Make the object invisible, then display a message so that
' the drawing is visible and users can see that the
' object is no longer visible.
polygon.Visible = False
polygon.Update
MsgBox ("...and now you don't.")

' Make the object visible again, so that users can manipulate
' it through the user interface.
polygon.Visible = True
polygon.Update

End Sub
Void Example

Sub Example_Void()

    'This example will display the if the ring is void or not

    Dim db As New AecBaseDatabase
    Dim ring As AecRing

    db.Init ThisDrawing.Database

    Set ring = db.ProfileStyles.Item(0).Profile.rings.Item(0)
    MsgBox "Ring Void?: " & ring.Void, vbInformation, "Void Example"

End Sub
Volume Example

Sub Example_Volume()

'This example shows the volume of a mass element or mass group.

    Dim object As Object
    Dim pt As Variant
    Dim mass As AecMassElement
    Dim massGroup As AecMassGroup
    Dim count As Integer
    ThisDrawing.Utility.GetEntity object, pt, "Select Mass Element or Mass Group"

    If TypeOf object Is AecMassGroup Then
        Set massGroup = object
        MsgBox "Mass Group volume = " & Format(massGroup.Volume, "###0.000")
    Else
        If TypeOf object Is AecMassElement Then
            Set mass = object
            MsgBox "Mass Element volume = " & Format(mass.Volume, "###0.000")
        Else
            MsgBox "Mass Element or Group not selected.", vbInformation, "Volume Example"
        End If
    End If

End Sub
VolumeDisplay Example

Sub Example_VolumeDisplayUnit()

' This example returns the VolumeDisplayUnit setting for the current drawing.

Dim dbPref As AecArchBaseDatabasePreferences

Set dbPref = AecArchBaseApplication.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"

End Select

Print unit
Case aecUnitCubicDecimeter
    unit = "cubic decimeters"

Case aecUnitCubicMeter
    unit = "cubic meters"

End Select

MsgBox "The current value for VolumeDisplayUnit is " & unit, _
         vbInformation, "VolumeDisplayUnit Example"

End Sub
Sub Example_VolumePrecision()

' This example displays the VolumePrecision setting for the current drawing.
Dim dbPref As AecArchBaseDatabasePreferences
Set dbPref = AecArchBaseApplication.ActiveDocument.preferences

MsgBox "The current value for VolumePrecision is: " & dbPref.VolumePrecision,
vbInformation, "VolumePrecision Example"

End Sub
Sub Example_VolumeSuffix()

' This example displays the VolumeSuffix setting for the current drawing.

Dim dbPref As AecArchBaseDatabasePreferences

Set dbPref = AecArchBaseApplication.ActiveDocument.preferences

MsgBox "The current value for VolumeSuffix is " & dbPref.VolumeSuffix,
       vbInformation, "VolumeSuffix Example"

End Sub
# Width Example

**Examples:**

- [AecLayoutGrid2D](#)
- [AecLayoutGrid3D](#)
- [AecMassElement](#)

---

**Sub** Example_Width_AecLayoutGrid2D()

'This example displays the width of a 2D layout grid

Dim obj As Object  
Dim pt As Variant  
Dim grid As AecLayoutGrid2D

ThisDrawing.Utility.GetEntity obj, pt, "Select a 2D Layout Grid"

If TypeOf obj Is AecLayoutGrid2D Then  
   Set grid = obj  
   MsgBox "Grid Width is: " & grid.Width, vbInformation, "Width Example"  
Else  
   MsgBox "Not a 2D Layout Grid", vbExclamation, "Width Example"  
End If

End Sub

---

**Sub** Example_Width_AecLayoutGrid3D()

'This example displays the width of a 3D layout grid

Dim obj As Object  
Dim pt As Variant  
Dim grid As AecLayoutGrid3D
ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

If TypeOf obj Is AecLayoutGrid3D Then
    Set grid = obj
    MsgBox "Grid Width is: " & grid.Width, vbInformation, "Width Example"
Else
    MsgBox "Not a 3D Layout Grid", vbExclamation, "Width Example"
End If

End Sub

Sub Example_Width_AecMassElement()

' This example shows the size of the mass element in its relative X direction

    Dim obj As Object
    Dim pt As Variant
    Dim mass As AecMassElement

    ThisDrawing.Utility.GetEntity obj, pt, "Select Mass Element"

    If TypeOf obj Is AecMassElement Then<
        /font >
        Set mass =
        obj MsgBox "Mass Element Width is: " & mass.Width, vbInformation,
    Else
        MsgBox "No Mass Elements selected.", vbInformation, "Width Example"
    End If

End Sub
WindowState Example

Sub Example_WindowState()
    ' This example reads and displays the current window state of the AutoCAD app

    Dim CurrentState As String

    ' Use the "WindowState" variable to determine the window state of AutoCAD
    Select Case WindowState
        Case acMin: CurrentState = "Minimized"
        Case acMax: CurrentState = "Maximized"
        Case acNorm: CurrentState = "Normal Size"
    End Select

    ' Display window state
    MsgBox "The application window is now: " & CurrentState
End Sub
WindowTitle Example

Sub Example_WindowTitle()

' This example cycles through the documents collection
' and uses the WindowTitle property to create a list of all open documents.

Dim DOC As AecBaseDocument
Dim aecApp As New AecBaseApplication
Dim msg As String

' Initialize the Aec Base application object
aecApp.Init ThisDrawing.Application

' If there are no open documents, then exit
If aecApp.Documents.count = 0 Then
    MsgBox "There are no open documents!"
    Exit Sub
End If

msg = vbCrLf & vbCrLf ' Start with a space

' Cycle through all open drawings and get the window title of each drawing
For Each DOC In aecApp.Documents
    msg = msg & DOC.WindowTitle & vbCrLf
Next

' Display a list of open drawings
MsgBox "The open drawing titles are: " & msg

End Sub
Sub Example_XDistance()

    ' This example will make a mass element and anchor it 200 units from the start of a line.

    Dim mass As AecMassElement
    Dim line As AcadLine
    Dim anchor As New AecAnchorEntToCurve
    Dim pt1(0 To 2) As Double
    Dim pt2(0 To 2) As Double

    pt2(0) = 1000: pt2(1) = 1000

    ' this makes a new mass element 120 x 120 x 120 at 0,0,0
    Set mass = ThisDrawing.ModelSpace.AddCustomObject("AecMassElement")
    ' draws a line from 0,0 to 1000,1000
    Set line = ThisDrawing.ModelSpace.AddLine(pt1, pt2)

    anchor.Reference = line ' set the line as the object (curve) to anchor to
    anchor.XDistance = 200 ' set the X Distance for the anchor
    mass.AttachAnchor anchor ' attach the anchor to the mass element

End Sub
XEndoffset Example

Examples:

1 AecLayoutGrid2D
1 AecLayoutGrid3D

Sub Example_XEndoffset_AecLayoutGrid2D()

' This example displays the X end offset for a 2D layout grid

Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid2D

ThisDrawing.Utility.GetEntity obj, pt, "Select a 2D Layout Grid"

If TypeOf obj Is AecLayoutGrid2D Then
    Set grid = obj
    MsgBox "Grid X End Offset is: " & grid.XEndOffset, vbInformation, "XEndOffset Example"
Else
    MsgBox "Not a 2D Layout Grid", vbExclamation, "XEndOffset Example"
End If

End Sub

Sub Example_XEndoffset_AecLayoutGrid3D()

' This example displays the X end offset for a 3D layout grid

Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid3D
ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

If TypeOf obj Is AecLayoutGrid3D Then
    Set grid = obj
    MsgBox "Grid X End Offset is: " & grid.XEndOffset, vbInformation, "XEndOffset Example"
Else
    MsgBox "Not a 3D Layout Grid", vbExclamation, "XEndOffset Example"
End If

End Sub
**XNodes Example**

**Examples:**

1. **AecLayoutGrid2D**

2. **AecLayoutGrid3D**

---

```vbnet
Sub Example_XNodes_AecLayoutGrid2D()

    'This example displays the number of X nodes of a 2D layout grid

    Dim obj As Object
    Dim pt As Variant
    Dim grid As AecLayoutGrid2D

    ThisDrawing.Utility.GetEntity obj, pt, "Select a 2D Layout Grid"

    If TypeOf obj Is AecLayoutGrid2D Then
        Set grid = obj
        MsgBox "Grid X Nodes is: " & grid.XNodes.Count, vbInformation, "XNodes Example"
    Else
        MsgBox "Not a 2D Layout Grid", vbExclamation, "XNodes Example"
    End If

End Sub
```

---

```vbnet
Sub Example_XNodes_AecLayoutGrid3D()

    'This example displays the number of X nodes of a 3D layout grid

    Dim obj As Object
    Dim pt As Variant
    Dim grid As AecLayoutGrid3D
```
ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

If TypeOf obj Is AecLayoutGrid3D Then
    Set grid = obj
    MsgBox "Grid X Nodes is: " & grid.XNodes.Count, vbInformation, "XNodes Example"
Else
    MsgBox "Not a 3D Layout Grid", vbExclamation, "XNodes Example"
End If

End Sub
This example shows the Xoffset of the first viewblock of a multiviewblock

Sub Example_XOffset()

ThisDrawing.Utility.GetEntity obj, pt, "Select a Multiview Block"
If TypeOf obj Is AecMVBlockRef Then
    Set blockRef = obj
    Set viewBlocks = blockRef.viewBlocks
    MsgBox "XOffset of View Block 1: " & viewBlocks.Item(0).XOffset, vbInformation, "XOffset Example"
Else
    MsgBox "Not a Multiview Block", vbInformation, "XOffset Example"
End If

End Sub
Sub Example_XPositionFrom()

    ' This example will make a mass element and anchor it 200 units 
    ' from the end of a line.

    Dim mass As AecMassElement
    Dim line As AcadLine
    Dim anchor As New AecAnchorEntToCurve
    Dim pt1(0 To 2) As Double
    Dim pt2(0 To 2) As Double

    pt2(0) = 1000: pt2(1) = 1000

    ' this makes a new mass element 120 x 120 x 120 at 0,0,0
    Set mass = ThisDrawing.ModelSpace.AddCustomObject("AecMassElement")
    ' draws a line from 0,0 to 1000,1000
    Set line = ThisDrawing.ModelSpace.AddLine(pt1, pt2)

    anchor.Reference = line ' set the line as the object (curve) to anchor to
    anchor.XDistance = 200 ' set the X Distance for the anchor
    anchor.XPositionFrom = aecCurvePositionEnd ' set the anchor to measure from
    mass.AttachAnchor anchor ' attach the anchor to the mass element

End Sub
Sub Example_XPositionTo()

    ' This example will make a mass element and anchor it's center 200 units from the end of a line.

    Dim mass As AecMassElement
    Dim line As AcadLine
    Dim anchor As New AecAnchorEntToCurve
    Dim pt1(0 To 2) As Double
    Dim pt2(0 To 2) As Double

    pt2(0) = 1000: pt2(1) = 1000

    ' this makes a new mass element 120 x 120 x 120 at 0,0,0
    Set mass = ThisDrawing.ModelSpace.AddCustomObject("AecMassElement")
    ' draws a line from 0,0 to 1000,1000
    Set line = ThisDrawing.ModelSpace.AddLine(pt1, pt2)

    anchor.Reference = line ' set the line as the object (curve) to anchor to
    anchor.XDistance = 200 ' set the X Distance for the anchor
    anchor.XPositionFrom = aecCurvePositionEnd ' set the anchor to measure from
    anchor.XPositionTo = aecEdgePositionCenter ' set the anchor to measure to the mass.
    mass.AttachAnchor anchor ' attach the anchor to the mass element

End Sub
XRefEdit Example

Sub Example_XRefEdit()
    ' This example reads and modifies the preference value which controls
    ' whether the current drawing can be edited in place when being
    ' referenced by another user.
    '
    ' When finished, this example resets the preference value back to
    ' it's original value.

    Dim ACADPref As AcadDatabasePreferences
    Dim originalValue As Variant, newValue As Variant

    ' Get the user preferences object
    Set ACADPref = ThisDrawing.preferences

    ' Read and display the original value
    originalValue = ACADPref.XRefEdit
    MsgBox "The XRefEdit preference is set to: " & originalValue

    ' Modify the XRefEdit preference by toggling the value
    ACADPref.XRefEdit = Not (ACADPref.XRefEdit)
    newValue = ACADPref.XRefEdit
    MsgBox "The XRefEdit preference has been set to: " & newValue

    ' Reset the preference back to it's original value
    '
    ' * Note: Comment out this last section to leave the change to
    ' this preference in effect
    ACADPref.XRefEdit = originalValue
    MsgBox "The XRefEdit preference was reset back to: " & originalValue
End Sub
XRefLayerVisibility Example

Sub Example_XRefLayerVisibility()
    ' This example reads and modifies the preference value which controls
    ' the visibility of xref-dependent layers and specifies if nested xref
    ' path changes are saved.
    '
    ' When finished, this example resets the preference value back to
    ' it's original value.
    Dim ACADPref As AcadDatabasePreferences
    Dim originalValue As Variant, newValue As Variant
    Dim ACADPref As AcadDatabasePreferences
    Dim originalValue As Variant, newValue As Variant

    ' Get the user preferences object
    Set ACADPref = ThisDrawing.preferences

    ' Read and display the original value
    originalValue = ACADPref.XRefLayerVisibility
    MsgBox "The XRefLayerVisibility preference is set to: " & originalValue

    ' Modify the XRefLayerVisibility preference by toggling the value
    ACADPref.XRefLayerVisibility = Not (ACADPref.XRefLayerVisibility)
    newValue = ACADPref.XRefLayerVisibility
    MsgBox "The XRefLayerVisibility preference has been set to: " & newValue

    ' Reset the preference back to it's original value
    '
    ' * Note: Comment out this last section to leave the change to
    ' this preference in effect
    ACADPref.XRefLayerVisibility = originalValue
    MsgBox "The XRefLayerVisibility preference was reset back to: " & originalValue
End Sub
XRotation Example

Sub Example_XRotation()

' This example will make a mass element and anchor it to a line, and rotate it around the X axis of the line 45 degrees

Dim mass As AecMassElement
Dim line As AcadLine
Dim anchor As New AecAnchorEntToCurve
Dim pt1(0 To 2) As Double
Dim pt2(0 To 2) As Double

pt2(0) = 1000: pt2(1) = 1000

' this makes a new mass element 120 x 120 x 120 at 0,0,0
Set mass = ThisDrawing.ModelSpace.AddCustomObject("AecMassElement")
' draws a line from 0,0 to 1000,1000
Set line = ThisDrawing.ModelSpace.AddLine(pt1, pt2)

anchor.Reference = line ' set the line as the object (curve) to anchor to
anchor.XDistance = 200 ' set the X Distance for the anchor
anchor.XRotation = Atn(1) ' Rotate by 45 degrees (Atn of 1 is 45 degrees in radians)
mass.AttachAnchor anchor ' attach the anchor to the mass element

End Sub
**XStartOffset Example**

**Examples:**

1. `AecLayoutGrid2D`
2. `AecLayoutGrid3D`

---

**Sub** `Example_XStartOffset_AecLayoutGrid2D()`

'This example displays the X start offset for a 2D layout grid

```vba
Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid2D

ThisDrawing.Utility.GetEntity obj, pt, "Select a 2D Layout Grid"

If TypeOf obj Is AecLayoutGrid2D Then
    Set grid = obj
    MsgBox "Grid X Start Offset is: " & grid.XStartOffset, vbInformation, "XStartOffset Example"
Else
    MsgBox "Not a 2D Layout Grid", vbExclamation, "XStartOffset Example"
End If
```

End Sub

---

**Sub** `Example_XStartOffset_AecLayoutGrid3D()`

'This example displays the X start offset for a 3D layout grid

```vba
Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid3D
```
ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

If TypeOf obj Is AecLayoutGrid3D Then
    Set grid = obj
    MsgBox "Grid X Start Offset is: " & grid.XStartOffset, vbInformation, "XSt
Else
    MsgBox "Not a 3D Layout Grid", vbExclamation, "XStartOffset Example"
End If

End Sub
XType Example

Examples:

1 AecLayoutGrid2D
1 AecLayoutGrid3D

Sub Example_XType_AecLayoutGrid2D()

    'This example displays the X type for a 2D layout grid

    Dim obj As Object
    Dim pt As Variant
    Dim grid As AecLayoutGrid2D

    ThisDrawing.Utility.GetEntity obj, pt, "Select a 2D Layout Grid"

    If TypeOf obj Is AecLayoutGrid2D Then
        Set grid = obj

        Select Case grid.XType
            Case 0:
                MsgBox "Grid X Type is: Manual Spacing", vbInformation, "XType Example"
            Case 1:
                MsgBox "Grid X Type is: Auto Spacing Even", vbInformation, "XType Example"
            Case 2:
                MsgBox "Grid X Type is: Auto Spacing Bay", vbInformation, "XType Example"
        End Select
    Else
        MsgBox "Not a 2D Layout Grid", vbExclamation, "XType Example"
    End If

End Sub
Sub Example_XType_AecLayoutGrid3D()

  'This example displays the X type for a 3D layout grid

  Dim obj As Object
  Dim pt As Variant
  Dim grid As AecLayoutGrid3D

  ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

  If TypeOf obj Is AecLayoutGrid3D Then
    Set grid = obj

        Select Case grid.XType
        Case 0:
          MsgBox "Grid X Type is: Manual Spacing", vbInformation, "XType Example"
        Case 1:
          MsgBox "Grid X Type is: Auto Spacing Even", vbInformation, "XType Example"
        Case 2:
          MsgBox "Grid X Type is: Auto Spacing Bay", vbInformation, "XType Example"
        End Select

  Else
    MsgBox "Not a 3D Layout Grid", vbExclamation, "XType Example"
  End If

End Sub
XValue Example

Examples:

1 AecLayoutGrid2D

1 AecLayoutGrid3D

Sub Example_XValue_AecLayoutGrid2D()

'This example displays the X value for a 2D layout grid

Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid2D

ThisDrawing.Utility.GetEntity obj, pt, "Select a 2D Layout Grid"

If TypeOf obj Is AecLayoutGrid2D Then
    Set grid = obj
    MsgBox "Grid X Value is: " & grid.XValue, vbInformation, "XValue Example"
Else
    MsgBox "Not a 2D Layout Grid", vbExclamation, "XValue Example"
End If

End Sub

Sub Example_XValue_AecLayoutGrid3D()

'This example displays the X value for a 3D layout grid

Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid3D
ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

If TypeOf obj Is AecLayoutGrid3D Then
    Set grid = obj
    MsgBox "Grid X Value is: " & grid.XValue, vbInformation, "XValue Example"
Else
    MsgBox "Not a 3D Layout Grid", vbExclamation, "XValue Example"
End If

End Sub
Sub Example_YAlignment()

' This example looks at the way a selected object is anchored in relation to the baseline of a grid assembly.

' Use this example with a drawing that contains a window assembly and one or more AEC objects attached to the assembly.

Dim ent As AcadEntity
Dim geo As AecGeo
Dim anchor As AecAnchor

Dim alignment As String

On Error Resume Next ' Handle errors in code.

' Prompt user to select an object.
ThisDrawing.Utility.GetEntity ent, pt, "Select object anchored to window assembly:"

' Make sure user selected an AEC object, and that the object is anchored to a grid assembly.
If ent Is Nothing Then
    MsgBox "Nothing was selected.", vbExclamation, "YAlignment Example"
ElseIf TypeOf ent Is AecGeo Then
    Set geo = ent

    ' Get the anchor the object is attached to.
    Set anchor = geo.GetAnchor
    On Error GoTo 0
    If anchor Is Nothing Then
        MsgBox "Selected object is not anchored.", vbExclamation, "YAlignment Example"
    ElseIf Not TypeOf anchor Is AecAnchorEntToGridAssembly Then
        MsgBox "Object is anchored, but not to a grid assembly.", vbExclamation
    Else

Select Case anchor.YAlignment
    Case aecInfillAlignCentered
        alignment = "Centered."
    Case aecInfillAlignFrontFlush
        alignment = "In front of the baseline."
    Case aecInfillAlignBackFlush
        alignment = "In back of the baseline."
    Case Else
        alignment = "Unknown"
End Select

    MsgBox "Y Alignment of object: " & alignment, vbInformation, "YAlignment Example"
End If
Else
    MsgBox "Object selected is not an AEC entity.", vbInformation, "YAlignment Example"
End If

End Sub
Sub Example_YDistance()

' This example will make a mass element and anchor it 200 units to the left of a line.

Dim mass As AecMassElement
Dim line As AcadLine
Dim anchor As New AecAnchorEntToCurve
Dim pt1(0 To 2) As Double
Dim pt2(0 To 2) As Double

pt2(0) = 1000: pt2(1) = 1000

' this makes a new mass element 120 x 120 x 120 at 0,0,0
Set mass = ThisDrawing.ModelSpace.AddCustomObject("AecMassElement")
' draws a line from 0,0 to 1000,1000
Set line = ThisDrawing.ModelSpace.AddLine(pt1, pt2)

anchor.Reference = line ' set the line as the object (curve) to anchor to
anchor.YDistance = 200 ' set the Y Distance for the anchor
mass.AttachAnchor anchor ' attach the anchor to the mass element

End Sub
**YEndOffset Example**

**Examples:**

- **AecLayoutGrid2D**
- **AecLayoutGrid3D**

---

**Sub** Example_YEndOffset_AecLayoutGrid2D()

'This example displays the Y end offset for a 2D layout grid

Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid2D

ThisDrawing.Utility.GetEntity obj, pt, "Select a 2D Layout Grid"

If TypeOf obj Is AecLayoutGrid2D Then
    Set grid = obj
    MsgBox "Grid Y End Offset is: " & grid.YEndOffset, vbInformation, "YEndOffset Example"
Else
    MsgBox "Not a 2D Layout Grid", vbExclamation, "YEndOffset Example"
End If

End Sub

---

**Sub** Example_YEndOffset_AecLayoutGrid3D()

'This example displays the Y end offset for a 3D layout grid

Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid3D
ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

    If TypeOf obj Is AecLayoutGrid3D Then
        Set grid = obj
        MsgBox "Grid Y End Offset is: " & grid.YEndOffset, vbInformation, "YEndOffset Example"
    Else
        MsgBox "Not a 3D Layout Grid", vbExclamation, "YEndOffset Example"
    End If

End Sub
YNodes Example

Examples:

1 AecLayoutGrid2D
1 AecLayoutGrid3D

Sub Example_YNodes_AecLayoutGrid2D()

'This example displays the number of Y nodes for a 2D layout grid

Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid2D

ThisDrawing.Utility.GetEntity obj, pt, "Select a 2D Layout Grid"

If TypeOf obj Is AecLayoutGrid2D Then
    Set grid = obj
    MsgBox "Grid Y Nodes is: " & grid.YNodes.Count, vbInformation, "YNodes Example"
Else
    MsgBox "Not a 2D Layout Grid", vbExclamation, "YNodes Example"
End If

End Sub

Sub Example_YNodes_AecLayoutGrid3D()

'This example displays the number of Y nodes for a 3D layout grid

Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid3D
ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

If TypeOf obj Is AecLayoutGrid3D Then
    Set grid = obj
    MsgBox "Grid Y Nodes is: " & grid.YNodes.Count, vbInformation, "YNodes Example"
Else
    MsgBox "Not a 3D Layout Grid", vbExclamation, "YNodes Example"
End If

End Sub
Sub Example_YOffset ()

'This example shows the Yoffset of the first viewblock of a multiviewblock

Dim obj As Object
Dim pt As Variant
Dim blockRef As AecMVBlockRef
Dim viewBlocks As AecViewBlocks

ThisDrawing.Utility.GetEntity obj, pt, "Select a Multiview Block"
If TypeOf obj Is AecMVBlockRef Then
    Set blockRef = obj
    Set viewBlocks = blockRef.viewBlocks
    MsgBox "YOffset of View Block 1: " & viewBlocks.Item(0).YOffset, vbInformation, "YOffset Example"
Else
    MsgBox "Not a Multiview Block", vbInformation, "YOffset Example"
End If

End Sub
YPositionFrom Example

Sub Example_YPositionFrom()

' This example will make a mass element and anchor it 200 units from the left of a line.

Dim mass As AecMassElement
Dim line As AcadLine
Dim anchor As New AecAnchorEntToCurve
Dim pt1(0 To 2) As Double
Dim pt2(0 To 2) As Double

pt2(0) = 1000: pt2(1) = 1000

' this makes a new mass element 120 x 120 x 120 at 0,0,0
Set mass = ThisDrawing.ModelSpace.AddCustomObject("AecMassElement")
' draws a line from 0,0 to 1000,1000
Set line = ThisDrawing.ModelSpace.AddLine(pt1, pt2)

anchor.Reference = line ' set the line as the object (curve) to anchor to
anchor.YDistance = 200 ' set the Y Distance for the anchor
anchor.YPositionFrom = aecCurvePositionEnd ' set the anchor to measure from
mass.AttachAnchor anchor ' attach the anchor to the mass element

End Sub
Sub Example_YPositionTo()

    ' This example will make a mass element and anchor it's center 200 units
    ' from the left of a line.

    Dim mass As AecMassElement
    Dim line As AcadLine
    Dim anchor As New AecAnchorEntToCurve
    Dim pt1(0 To 2) As Double
    Dim pt2(0 To 2) As Double

    pt2(0) = 1000: pt2(1) = 1000

    ' this makes a new mass element 120 x 120 x 120 at 0,0,0
    Set mass = ThisDrawing.ModelSpace.AddCustomObject("AecMassElement")
    ' draws a line from 0,0 to 1000,1000
    Set line = ThisDrawing.ModelSpace.AddLine(pt1, pt2)

    anchor.Reference = line ' set the line as the object (curve) to anchor to
    anchor.YDistance = 200 ' set the Y Distance for the anchor
    anchor.YPositionTo = aecEdgePositionCenter ' set the anchor to measure to the
    mass.AttachAnchor anchor ' attach the anchor to the mass element

End Sub
Sub Example_YRotation()

    'This example will add anchor a new mass element to a 2D layout grid in the ' drawing.

    Dim grid As AecLayoutGrid2D
    Dim mass As AecMassElement
    Dim pt As Variant
    Dim obj As AcadObject

    ThisDrawing.Utility.GetEntity obj, pt, "Select grid to attach to"
    If TypeOf obj Is AecLayoutGrid2D Then
        Set grid = obj
        Set mass = ThisDrawing.ModelSpace.AddCustomObject("AecMassElement"
        Dim anchor As New AecAnchorEntToLayoutNode
        anchor.Reference = grid
        ' anchor the mass element to the first node on the grid
        anchor.Node = 1
        anchor.YRotation = Atn(1) ' 45 degrees
        mass.AttachAnchor anchor
    Else
        MsgBox "No Layout Grid selected", vbInformation, "Node Example"
    End If

End Sub
Examples:

1 AecLayoutGrid2D
1 AecLayoutGrid3D

Sub Example_YStartOffset_AecLayoutGrid2D()

' This example displays the Y start offset for a 2D layout grid

Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid2D

ThisDrawing.Utility.GetEntity obj, pt, "Select a 2D Layout Grid"

If TypeOf obj Is AecLayoutGrid2D Then
    Set grid = obj
    MsgBox "Grid Y Start Offset is: " & grid.YStartOffset, vbInformation, "YStartOffset Example"
Else
    MsgBox "Not a 2D Layout Grid", vbExclamation, "YStartOffset Example"
End If

End Sub

Sub Example_YStartOffset_AecLayoutGrid3D()

' This example displays the Y start offset for a 3D layout grid

Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid3D
ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

If TypeOf obj Is AecLayoutGrid3D Then
    Set grid = obj
    MsgBox "Grid Y Start Offset is: " & grid.YStartOffset, vbInformation, "YSt"
Else
    MsgBox "Not a 3D Layout Grid", vbExclamation, "YStartOffset Example"
End If

End Sub
**YType Example**

**Examples:**

1 AecLayoutGrid2D
1 AecLayoutGrid3D

---

**Sub** Example_YType_AecLayoutGrid2D()

'This example displays the Y type for a 2D layout grid

Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid2D

ThisDrawing.Utility.GetEntity obj, pt, "Select a 2D Layout Grid"

If TypeOf obj Is AecLayoutGrid2D Then
  Set grid = obj

  Select Case grid.YType
  Case 0:
    MsgBox "Grid Y Type is: Manual Spacing", vbInformation, "YType Example"
  Case 1:
    MsgBox "Grid Y Type is: Auto Spacing Even", vbInformation, "YType Example"
  Case 2:
    MsgBox "Grid Y Type is: Auto Spacing Bay", vbInformation, "YType Example"
  End Select

Else
  MsgBox "Not a 2D Layout Grid", vbExclamation, "YType Example"
End If

End Sub
Sub Example_YType_AecLayoutGrid3D()

' This example displays the Y type for a 3D layout grid

Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid3D

ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

If TypeOf obj Is AecLayoutGrid3D Then
    Set grid = obj

    Select Case grid.YType
    Case 0:
        MsgBox "Grid Y Type is: Manual Spacing", vbInformation, "YType Example"
    Case 1:
        MsgBox "Grid Y Type is: Auto Spacing Even", vbInformation, "YType Example"
    Case 2:
        MsgBox "Grid Y Type is: Auto Spacing Bay", vbInformation, "YType Example"
    End Select

Else
    MsgBox "Not a 3D Layout Grid", vbExclamation, "YType Example"
End If

End Sub
YValue Example

Examples:

1 AecLayoutGrid2D
1 AecLayoutGrid3D

Sub Example_YValue_AecLayoutGrid2D()

    "This example displays the Y value for a 2D layout grid"

    Dim obj As Object
    Dim pt As Variant
    Dim grid As AecLayoutGrid2D

    ThisDrawing.Utility.GetEntity obj, pt, "Select a 2D Layout Grid"

    If TypeOf obj Is AecLayoutGrid2D Then
        Set grid = obj
        MsgBox "Grid Y Value is: " & grid.YValue, vbInformation, "YValue Example"
    Else
        MsgBox "Not a 2D Layout Grid", vbExclamation, "YValue Example"
    End If

End Sub

Sub Example_YValue_AecLayoutGrid3D()

    "This example displays the Y value for a 3D layout grid"

    Dim obj As Object
    Dim pt As Variant
    Dim grid As AecLayoutGrid3D
ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

If TypeOf obj Is AecLayoutGrid3D Then
    Set grid = obj
    MsgBox "Grid Y Value is: " & grid.YValue, vbInformation, "YValue Example"
Else
    MsgBox "Not a 3D Layout Grid", vbExclamation, "YValue Example"
End If

End Sub
Sub Example_ZDistance()

' This example will make a mass element and anchor it 200 units above a line.

Dim mass As AecMassElement
Dim line As AcadLine
Dim anchor As New AecAnchorEntToCurve
Dim pt1(0 To 2) As Double
Dim pt2(0 To 2) As Double

pt2(0) = 1000: pt2(1) = 1000

' this makes a new mass element 120 x 120 x 120 at 0,0,0
Set mass = ThisDrawing.ModelSpace.AddCustomObject("AecMassElement")
' draws a line from 0,0 to 1000,1000
Set line = ThisDrawing.ModelSpace.AddLine(pt1, pt2)

anchor.Reference = line ' set the line as the object (curve) to anchor to
anchor.ZDistance = 200 ' set the Z Distance for the anchor
mass.AttachAnchor anchor ' attach the anchor to the mass element

End Sub
Sub Example_ZEndOffset()

    'This example displays the Z end offset for a 3D layout grid

    Dim obj As Object
    Dim pt As Variant
    Dim grid As AecLayoutGrid3D

    ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

    If TypeOf obj Is AecLayoutGrid3D Then
        Set grid = obj
        MsgBox "Grid Z End Offset is: " & grid.ZEndOffset, vbInformation, "ZEndOffset Example"
    Else
        MsgBox "Not a 3D Layout Grid", vbExclamation, "ZEndOffset Example"
    End If

End Sub
ZNodes Example

Sub Example_ZNodes()

    'This example displays the number of Z nodes for a 3D layout grid

    Dim obj As Object
    Dim pt As Variant
    Dim grid As AecLayoutGrid3D

    ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

    If TypeOf obj Is AecLayoutGrid3D Then
        Set grid = obj
        MsgBox "Grid Z Nodes is: " & grid.ZNodes.Count, vbInformation, "ZNodes Example"
    Else
        MsgBox "Not a 3D Layout Grid", vbExclamation, "ZNodes Example"
    End If

End Sub
Sub Example_ZOffset ()

'This example shows the Zoffset of the first viewblock of a multiviewblock

Dim obj As Object
Dim pt As Variant
Dim blockRef As AecMVBlockRef
Dim viewBlocks As AecViewBlocks

ThisDrawing.Utility.GetEntity obj, pt, "Select a Multiview Block"
If TypeOf obj Is AecMVBlockRef Then
    Set blockRef = obj
    Set viewBlocks = blockRef.viewBlocks
    MsgBox "Zoffset of View Block 1: " & viewBlocks.Item(0).ZOffset, vbInformation, "ZOffset Example"
Else
    MsgBox "Not a Multiview Block", vbInformation, "ZOffset Example"
End If

End Sub
Sub Example_Zoom()

' This example displays the zoom of a selected AecCamera object

Dim obj As Object
Dim pt As Variant
Dim camera As AecCamera

ThisDrawing.Utility.GetEntity obj, pt, "Select a Camera"

If TypeOf obj Is AecCamera Then
    Set camera = obj
    MsgBox "Zoom is: " & camera.Zoom, vbInformation, "Zoom Example"
Else
    MsgBox "Not a Camera", vbExclamation, "Zoom Example"
End If

End Sub
Sub Example_ZPositionFrom()

' This example will make a mass element and anchor it 200 units above a line.

Dim mass As AecMassElement
Dim line As AcadLine
Dim anchor As New AecAnchorEntToCurve
Dim pt1(0 To 2) As Double
Dim pt2(0 To 2) As Double

pt2(0) = 1000: pt2(1) = 1000

' this makes a new mass element 120 x 120 x 120 at 0,0,0
Set mass = ThisDrawing.ModelSpace.AddCustomObject("AecMassElement")
' draws a line from 0,0 to 1000,1000

Set line = ThisDrawing.ModelSpace.AddLine(pt1, pt2)

anchor.Reference = line ' set the line as the object (curve) to anchor to
anchor.ZDistance = -200 ' set the Z Distance for the anchor
anchor.ZPositionFrom = aecCurvePositionEnd ' set the anchor to measure from
mass.AttachAnchor anchor ' attach the anchor to the mass element

End Sub
Sub Example_ZPositionTo()

' This example will make a mass element and anchor it's center 200 units above a line.

Dim mass As AecMassElement
Dim line As AcadLine
Dim anchor As New AecAnchorEntToCurve
Dim pt1(0 To 2) As Double
Dim pt2(0 To 2) As Double

pt2(0) = 1000: pt2(1) = 1000

' this makes a new mass element 120 x 120 x 120 at 0,0,0
Set mass = ThisDrawing.ModelSpace.AddCustomObject("AecMassElement")
' draws a line from 0,0 to 1000,1000
Set line = ThisDrawing.ModelSpace.AddLine(pt1, pt2)

anchor.Reference = line ' set the line as the object (curve) to anchor to
anchor.ZDistance = -200 ' set the Z Distance for the anchor
anchor.ZPositionFrom = aecCurvePositionEnd ' set the anchor to measure from:
anchor.ZPositionTo = aecEdgePositionCenter ' set the anchor to measure to the
mass.AttachAnchor anchor ' attach the anchor to the mass element

End Sub
Sub Example_ZRotation()

' This example will make a mass element and anchor it to a line, and rotate it around the Z axis of the line 45 degrees

Dim mass As AecMassElement
Dim line As AcadLine
Dim anchor As New AecAnchorEntToCurve
Dim pt1(0 To 2) As Double
Dim pt2(0 To 2) As Double

pt2(0) = 1000: pt2(1) = 1000

' this makes a new mass element 120 x 120 x 120 at 0,0,0
Set mass = ThisDrawing.ModelSpace.AddCustomObject("AecMassElement")
' draws a line from 0,0 to 1000,1000
Set line = ThisDrawing.ModelSpace.AddLine(pt1, pt2)

anchor.Reference = line ' set the line as the object (curve) to anchor to
anchor.ZDistance = 200 ' set the Z Distance for the anchor
anchor.ZRotation = Atn(1) ' Rotate by 45 degrees (Atn of 1 is 45 degrees in rad
mass.AttachAnchor anchor ' attach the anchor to the mass element

End Sub
Sub Example_ZStartOffset()

    'This example displays the Z start offset for a 3D layout grid

    Dim obj As Object
    Dim pt As Variant
    Dim grid As AecLayoutGrid3D

    ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

    If TypeOf obj Is AecLayoutGrid3D Then
        Set grid = obj
        MsgBox "Grid Z Start Offset is: " & grid.ZStartOffset, vbInformation, "ZStartOffset Example"
    Else
        MsgBox "Not a 3D Layout Grid", vbExclamation, "ZStartOffset Example"
    End If

End Sub
Sub Example_ZType()

    'This example displays the Z type for a 3D layout grid

    Dim obj As Object
    Dim pt As Variant
    Dim grid As AecLayoutGrid3D

    ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

    If TypeOf obj Is AecLayoutGrid3D Then
        Set grid = obj

        Select Case grid.ZType
            Case 0:
                MsgBox "Grid Z Type is: Manual Spacing", vbInformation, "ZType Example"
            Case 1:
                MsgBox "Grid Z Type is: Auto Spacing Even", vbInformation, "ZType Example"
            Case 2:
                MsgBox "Grid Z Type is: Auto Spacing Bay", vbInformation, "ZType Example"
        End Select
    Else
        MsgBox "Not a 3D Layout Grid", vbExclamation, "ZType Example"
    End If

End Sub
Sub Example_ZValue()

' This example displays the Z value for a 3D layout grid

Dim obj As Object
Dim pt As Variant
Dim grid As AecLayoutGrid3D

ThisDrawing.Utility.GetEntity obj, pt, "Select a 3D Layout Grid"

If TypeOf obj Is AecLayoutGrid3D Then
    Set grid = obj
    MsgBox "Grid Z Value is: " & grid.ZValue, vbInformation, "ZValue Example"
Else
    MsgBox "Not a 3D Layout Grid", vbExclamation, "ZValue Example"
End If

End Sub
**ForceHorizontal Property**

Specifies whether the tag is aligned with the object or oriented horizontally.

**Signature**

```plaintext
object.ForceHorizontal
```

- **object**
  - AecAnchorExtendedTagToEnt: The object this property applies to.
- **ForceHorizontal**
  - Boolean; read-write
  - TRUE: Orient the tag horizontally.
  - FALSE: Align the tag with the object.

See Also | Example
**TrueColor Property**

Returns the true color of the object.

**Signature**

```
object.TrueColor
```

object


The object or objects this property applies to.

**TrueColor**

```
AcadAcCmColor object; read-only
```

AutoCAD True Color object.
**ActiveConfiguration Property**

Returns the active display configuration for the specified view port.

See Also | Example

**Signature**

RetVal = object.ActiveConfiguration(ViewPort)

**object**

*AecDisplayConfigurations* The object this property applies to.

**ViewPort**

*AcadObject; input-only*

The AcadViewport or AcadPViewport that the display configuration applies to.

**RetVal**

*AecDisplayConfiguration*

The active display configuration for the specified viewport.
AddOverride Method

Adds an object or style override on the display representation.

Signature

RetVal = object.AddOverride(AecObject)

Object

AecDisplayRepresentation The object or objects this method applies to.

AecObject

AecObject; input-only
An object instance or style override.

RetVal

AecDisplayProperties
The override display properties for the display representation.

Remarks

This method will fail if an override is already attached.
DeleteOverride Method

Deletes an existing override from the display representation.

See Also | Example

Signature

object.DeleteOverride AecObject

Object

AecDisplayRepresentation The object or objects this method applies to.

AecObject

AecObject; input-only
An object instance or style override.

Remarks

This method will fail if there is no existing override to remove.
**Duplicate Method**

Creates a duplicate display representation.

**Signature**

RetVal = object.Duplicate UniqueName

_object_ `AecDisplayRepresentation` The object or objects this method applies to.

_UniqueName_ `String; input-only`
The unique name for the new display representation.

RetVal `AecDisplayRepresentation`
The duplicated display representation.

**Remarks**

This method will fail if the name is not unique.
See Also

Methods and Properties:

ImportFreeForm method
See Also

Methods and Properties:

- ExportFreeForm method
Collections

AecLayerKeys
AecLayerKeyStyles
AecLayerOverrideSettings
AecLayoutCurveNodes
AecLayoutGrid2DNodes
AecLayoutGrid2DXNodes
AecLayoutGrid2DYNodes
AecLayoutGrid3DNodes
AecLayoutGrid3DXNodes
AecLayoutGrid3DYNodes
AecLayoutGrid3DZNodes
AecLayoutNodes
AecMaskBlockStyles
AecMassElementStyles
AecMassGroups
AecMVBlockStyles
AecPolygonStyles
AecProfileStyles
AecRings
AecViewBlocks
ACAD_ANGLE data type

ACAD_ANGLE is a DOUBLE whose units are based on the AUNITS system variable, and whose precision is based on the AUPREC system variable.
See Also:

Methods and Properties:

AngleToString method

AutoCAD ActiveX and VBA Developer's Guide:

Calculating Points and Values
See Also:

Methods and Properties:

AngleToString method

AutoCAD ActiveX and VBA Developer's Guide:

Calculating Points and Values
See Also:

Methods and Properties:

`AngleToString` method

AutoCAD ActiveX and VBA Developer's Guide:

Calculating Points and Values
The view direction for a given display set.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aecViewDirectionTop</td>
<td>0</td>
<td>Top</td>
</tr>
<tr>
<td>aecViewDirectionBottom</td>
<td>1</td>
<td>Bottom</td>
</tr>
<tr>
<td>aecViewDirectionLeft</td>
<td>2</td>
<td>Left</td>
</tr>
<tr>
<td>aecViewDirectionRight</td>
<td>3</td>
<td>Right</td>
</tr>
<tr>
<td>aecViewDirectionFront</td>
<td>4</td>
<td>Front</td>
</tr>
<tr>
<td>aecViewDirectionBack</td>
<td>5</td>
<td>Back</td>
</tr>
<tr>
<td>aecViewDirectionModel</td>
<td>6</td>
<td>Model</td>
</tr>
</tbody>
</table>
The fixed view direction used by a display configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aecFixedViewDirectionTop</td>
<td>0</td>
<td>Top</td>
</tr>
<tr>
<td>aecFixedViewDirectionBottom</td>
<td>1</td>
<td>Bottom</td>
</tr>
<tr>
<td>aecFixedViewDirectionLeft</td>
<td>2</td>
<td>Left</td>
</tr>
<tr>
<td>aecFixedViewDirectionRight</td>
<td>3</td>
<td>Right</td>
</tr>
<tr>
<td>aecFixedViewDirectionFront</td>
<td>4</td>
<td>Front</td>
</tr>
<tr>
<td>aecFixedViewDirectionBack</td>
<td>5</td>
<td>Back</td>
</tr>
<tr>
<td>aecFixedViewDirectionViewport</td>
<td>6</td>
<td>Viewport</td>
</tr>
</tbody>
</table>
The hatch type used by a display component.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aecHatchTypeUserDefined</td>
<td>0</td>
<td>UserDefined</td>
</tr>
<tr>
<td>aecHatchTypePreDefined</td>
<td>1</td>
<td>PreDefined</td>
</tr>
<tr>
<td>aecHatchTypeCustomDefined</td>
<td>2</td>
<td>CustomDefined</td>
</tr>
<tr>
<td>aecHatchTypeSolidFill</td>
<td>3</td>
<td>SolidFill</td>
</tr>
</tbody>
</table>
The surface hatch placement.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aecSurfaceHatchPlacementTop</td>
<td>1</td>
<td>Top</td>
</tr>
<tr>
<td>aecSurfaceHatchPlacementBottom</td>
<td>2</td>
<td>Bottom</td>
</tr>
<tr>
<td>aecSurfaceHatchPlacementLeft</td>
<td>3</td>
<td>Left</td>
</tr>
<tr>
<td>aecSurfaceHatchPlacementRight</td>
<td>4</td>
<td>Right</td>
</tr>
<tr>
<td>aecSurfaceHatchPlacementFront</td>
<td>5</td>
<td>Front</td>
</tr>
<tr>
<td>aecSurfaceHatchPlacementBack</td>
<td>6</td>
<td>Back</td>
</tr>
</tbody>
</table>
The surface mapping type.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aecSurfaceMappingTypeDefaultMapping</td>
<td>0</td>
<td>DefaultMapping</td>
</tr>
<tr>
<td>aecSurfaceMappingTypeFaceMapping</td>
<td>1</td>
<td>FaceMapping</td>
</tr>
<tr>
<td>aecSurfaceMappingTypeAsSurfaceHatch</td>
<td>2</td>
<td>AsSurfaceHatch</td>
</tr>
</tbody>
</table>
ACAD\_DISTANCE data type

ACAD\_DISTANCE is a DOUBLE whose units are based on the LUNITS system variable, and whose precision is based on the LUPREC system variable.