AutoCAD Map 3D 2009 ActiveX Reference

080207

The AutoCAD Map 3D ActiveX API covers data extension functionality (drawing sets, queries, object data... the "ADE" functionality).

For broader coverage of AutoCAD Map 3D functionality, use the .NET API, which is included in the AutoCAD Map 3D SDK. The SDK is a separate installation from AutoCAD Map 3D itself. You can download it from the Autodesk web site.

Note The ActiveX API remains available to support legacy applications that use it. For all new development, the .NET API is the best choice.

To download the AutoCAD Map 3D SDK

- 1. Open the <u>Autodesk Home Page</u> (it opens in a new window) and click Search.
- 2. Search for "ObjectARX for Autodesk Map 3D".
- 3. Click "Developer Center ObjectARX for Autodesk Map 3D".
- 4. Click "License And Download".

ap 3D 2009 ActiveX Reference

3D ActiveX API covers data extension functionality s, object data... the "ADE" functionality).

e of AutoCAD Map 3D functionality, use the .NET API, the AutoCAD Map 3D SDK. The SDK is a separate toCAD Map 3D itself. You can download it from the

API remains available to support legacy applications that use opment, the .NET API is the best choice.

utoCAD Map 3D SDK

<u>sk Home Page</u> (it opens in a new window) and click Search. ctARX for Autodesk Map 3D". r Center - ObjectARX for Autodesk Map 3D". ...

What's New

No changes have been made to ActiveX for AutoCAD Map 3D 2009.

en made to ActiveX for AutoCAD Map 3D 2009.

Container relationships: AcadMap object

Automation objects listed

AcadMap object

<u>Alias object</u>

Aliases collection

AllBound object

AlterLine object

AlterLines collection

AttachedDrawing object

BufferFenceBound collection

BufferPolylineBound collection

CircleBound object

ClosedPolylineBound collection

DiagParam object

DrawingSet collection

ErrorEntry collection

ErrorStack collection

Expression object

FenceBound collection

HatchDef object

MapUtil object

NestedDrawings collection

ODFieldDef object

ODFieldDefs collection

ODFieldValue object

ODRecord collection

ODRecords collection

ODTable object

ODTables collection

Point3d object

PolygonBound collection

PolylineBound collection

Project object

ProjectOptions object

Projects collection

Query object

QueryBranch collection

QueryCategories collection

<u>QueryCategory collection</u>

<u>QueryLeaf object</u>

QueryReport collection

Range object

RangeTable collection

RangeTables collection

SavedQuery object

SaveSet collection

SystemOptions object

TextDef object

WindowBound object

Methods and Properties listed

Click <u>here</u> to see the list of properties.

Methods

Aliases.Add Aliases.FindByPath Aliases.Item Aliases.Remove AlterLines.Add AlterLines.AddHatchDef AlterLines.AddTextDef AlterLines.Item AlterLines.Remove AlterLines.RemoveAll AttachedDrawing.AddAllToSSet AttachedDrawing.CountInSSet AttachedDrawing.GetTableList AttachedDrawing.QuickView AttachedDrawing.RemoveAllFromSSet AttachedDrawing.Save AttachedDrawing.SaveNewObjs AttachedDrawing.SaveObjectsByArea AttachedDrawing.SaveObjectsByLayer AttachedDrawing.SaveQueriedObjects BufferFenceBound.Add BufferFenceBound.Item BufferFenceBound.Remove BufferPolylineBound.Item **ClosedPolylineBound.Item**

DrawingSet.Add DrawingSet.GetTableList **DrawingSet.Item** DrawingSet.Remove DrawingSet.ZoomExtents **ErrorEntry.Add ErrorEntry.Item** ErrorStack.Add ErrorStack.Item ErrorStack.RemoveAll **Expression.Execute Expression.Execute** FenceBound.Add FenceBound.Item FenceBound.Remove MapUtil.NewBufferFence MapUtil.NewBufferPolyline MapUtil.NewCircle MapUtil.NewClosedPolyline MapUtil.NewExpression MapUtil.NewFence MapUtil.NewLocationAll MapUtil.NewODFieldDefs MapUtil.NewPoint3d MapUtil.NewPolygon MapUtil.NewPolyline MapUtil.NewWindow NestedDrawings.Item

ODFieldDefs.Add ODFieldDefs.Item ODFieldDefs.Remove ODRecord.AttachTo ODRecord.Item **ODRecords.Init ODRecords.Next** ODRecords.Record **ODRecords.Remove ODRecords.Rewind ODRecords.Update ODTable.CreateRecord ODTable.GetODRecords ODTables.Add ODTables.GetODRecords ODTables.Item** PolygonBound.Add PolygonBound.Item PolygonBound.Remove PolylineBound.Item Project.RunExternalQuery Project.WhoLocksObject **Projects.Item** Query.Clear **Query.Define Query.Execute** Query.Load **Query.Save**

QueryBranch.Add QueryBranch.Clear QueryBranch.Item QueryBranch.Remove QueryCategories.Add **QueryCategories.Item QueryCategories.Remove** QueryCategory.Add **QueryCategory.AddFromFile** QueryCategory.Item QueryCategory.Remove QueryLeaf.SetDataCond QueryLeaf.SetLocationCond **QueryLeaf.SetPropertyCond** QueryLeaf.SetSQLCond QueryReport.Add QueryReport.Clear QueryReport.Item QueryReport.Remove RangeTable.Add RangeTable.Item RangeTable.Remove **RangeTables.Add** RangeTables.Item **RangeTables.Remove** SaveSet.AddObjects SaveSet.GetObjects SaveSet.RemoveObjects

Click <u>here</u> to see the list of methods.

Properties

AcadMap.Aliases property AcadMap.ErrorStack property AcadMap.Projects property AcadMap.SystemOptions property Alias.Directory property Alias.Name property Aliases.Count property Alterline.Property property Alterline.Value property Alterlines.Count property AttachedDrawing.Active property AttachedDrawing.AliasPath property AttachedDrawing.ApplyTransform property AttachedDrawing.Description property AttachedDrawing.Extents property AttachedDrawing.HasLocks property AttachedDrawing.HasNested property AttachedDrawing.IsTopLevel property AttachedDrawing.NestedDrawings property AttachedDrawing.Parent property AttachedDrawing.Projection property AttachedDrawing.SaveBackExt property AttachedDrawing.Trotate property AttachedDrawing.Tscale property AttachedDrawing.TXOffset property

AttachedDrawing.TYOffset property BufferFenceBound.Count property BufferFenceBound.Width property BufferPolylineBound.Count property BufferPolylineBound.ObjectID property BufferPolylineBound.Width property CircleBound.CenterX property **CircleBound.CenterY property** CircleBound.CenterZ property **CircleBound.Radius property** ClosedPolylineBound.Count property ClosedPolylineBound.ObjectID property **DiagParam.Message property DiagParam.Position property DiagParam.Source property** DiagParam.SqlStatement property DrawingSet.Count property DrawingSet.Extents property **ErrorEntry.Count property** ErrorEntry.ErrCode property **ErrorEntry.ErrMessage property ErrorEntry.ErrType property** ErrorStack.Count property ErrorStack.LastErrCode property **Expression.Value property** FenceBound.Count property HatchDef.Color property HatchDef.Layer property

HatchDef.Pattern property HatchDef.Property property HatchDef.Rotation property HatchDef.ScaleFactor property NestedDrawings.Count property **ODFieldDef.Default property** ODFieldDef.Description property **ODFieldDef.IsUpdatable property ODFieldDef.Name property ODFieldDef.Type property ODFieldDefs.Count property ODFieldDefs.IsUpdatable property ODFieldValue.Type property ODFieldValue.Value property ODRecord.Count property ODRecord.ObjectID property ODRecord.TableName property ODRecords.IsDone property ODRecords.OwnerDbID property ODTable.Description property ODTable.Name property ODTable.ODFieldDefs property ODTable.StoreAsXdata property ODTables.Count property** Point3d.X property Point3d.Y property Point3d.Z property PolygonBound.Count property

PolylineBound.Count property PolylineBound.ObjectID property Project.CurrentProjection property **Project.CurrQuery property** Project.DrawingSet property Project.IsMapActiveCommand property Project.MapUtil property **Project.ODTables property** Project.ProjectOptions property Project.QueryCategories property **Project.RangeTables property Project.SaveSet property Projects.Count property** Query.AlterDefined property Query.AlterProp property Query.AlterTag property **Query.Mode** property **Query.QueryBranch** property Query.Report property Query.ReportDefined property Query.StringContext property QueryBranch.Count property QueryBranch.JoinOp property QueryBranch.Type property QueryCategories.Count property QueryCategory.Count property QueryCategory.Name property QueryLeaf.DataType property

QueryLeaf.JoinOp property **QueryLeaf.LocationType property** QueryLeaf.Not property **QueryLeaf.Operation property** QueryLeaf.Source property **OueryLeaf.Type** property QueryLeaf.Value property **QueryReport.Count property** QueryReport.IsReportNested property QueryReport.IsTransformEnabled property QueryReport.ReportFileName property Range.Expression property **Range.Operation property** Range.ReturnValue property RangeTable.Count property **RangeTable.Description property** RangeTable.Name property **RangeTables.Count property** SavedQuery.Category property SavedQuery.Description property SavedQuery.IsExternal property SavedQuery.Name property SavedQuery.Path property SaveSet.Count property SaveSet.CountErased property SaveSet.CountLocked property SaveSet.CountNew property TextDef.Color property

TextDef.Height property TextDef.InsertPoint property TextDef.Justification property TextDef.Layer property TextDef.Property property TextDef.Rotation property TextDef.Rotation property TextDef.TextStyle property TextDef.TextValue property WindowBound.BottomLeft property

Constant groups listed

EAdeClassId

EAdeDwgStatus

EAdeMode

EAdeNotification

EAlterationType

<u>EClassId</u>

EConditionOperator

EDataQueryType

EDataType

<u>EDwgUpdateStatus</u>

<u>EErrCode</u>

EErrType

EJoinOperator

ELocationType

<u>EPrefType</u>

<u>EPropertyType</u>

<u>EQueryType</u>

ERangeOperator

ESaveQueryOptions

<u>ESaveSetObjectType</u>

ETableType

EUserRights

ıps listed

2

EErrCode

 See Also

 -15
 1200
 2000

 0
 1300
 2100

<u>200</u>	<u>1500</u>	<u>2200</u>	<u>2700</u>	<u>3200</u>
<u>1000</u>	<u>1800</u>	<u>2300</u>	<u>2800</u>	<u>4000</u>
<u>1100</u>	<u>1900</u>	<u>2400</u>	<u>2900</u>	

The following EErrCode constants correspond to the AcMap::EErrCode enumerators in the ObjectARX API and to the error codes in the ADSRX/AutoLISP API. Errors on the error stack are represented by ErrorEntry collections. Error codes are returned by the ErrorEntry.ErrCode property.

<u>3000</u>

<u>3100</u>

<u>2500</u>

2600

Errors are listed below by number. To see them listed alphabetically, choose EErrCode in the VBA object browser with the AutocadMap library selected. **Expression**

-15	kErrXEDValueFail	Internal error
-14	kErrRngTabEvalFail	Internal error
-13	kErrNoRngTabFound	Internal error
-12	kErrNoRngTabLibFound	Internal error
-11	kErrLispValueFail	Internal error
-10	kErrGetAttrFail	Internal error
-09	kErrGetSQLFail	Internal error
-08	kErrGetEEDFail	Internal error
-07	kErrInvalidProperty	Internal error

-06	kErrExpInvalidOperand	Internal error
-05	kErrExpADS	Internal error
-04	kErrExpNoMemforOperand	Internal error
-03	kErrExpMathOverFlow	Internal error
-02	kErrExpInvalidOper	Internal error
-01	kErrExpSyntaxErr	Internal error

For more Expression errors, which begin at 2900, click **≥**. **Common Usage**

00	kAdeOk	General ADE return value: call to ADE object is successful
01	kAdeErr	General ADE error: call to ADE object is not successful
02	kAdeBadInput	Invalid function argument
03	kAdeObjectNotFound	
04	kAdeOutOfMemory	
05	kAdeObjNotInitialized	
06	kAdeWrongType	
07	kAdeWrongProject	
80	kAdeEOB	
09	kAdeADSError	Invalid function argument type

- 10 kAdeAdsNameConvertionFails
- 11 kAdeWrongArgument
- 12 kAdeWriteBufFails
- 13 kAdeReadBufFails
- 14 kAdeXDataCorrupted
- 15 kAdeNoEnvironment
- 16 kAdeUsrBreak
- 17 kAdeUncomparable
- 18 kAdePermissionDenied

No permission to perform some action. When using the following commands, the message has different meanings: ADESAVEOBJS = can't be executed in demonstration mode: ADEDRAWINGS = User has no rights to update the set; ADESELOBJS, ADDREMOBJS = User has no rights to edit; ADEDWGMAINT = User needs to be superuser to remove the foreign locks; **MAPOPTIONS:** System Preferences = can't set "Force User Login" flag and change "Object Locking" flag if drawing set contains active or locked drawings. Deactivate

or unlock the drawings before trying to set these preferences.

- 19 kAdeWrongSyntax
- 20 kAdeDuplicate
- 21 kAdeInvalidPathOrFileName
- 22 kAdeInvalidVersion
- 23 kAdeFileIOFatalError

External Subsystems

- 200 kAdeAcDbError
- 201 kAdeIRDNotInitialized
- 202 kAdeIRDError
- 203 kAdeASENotInitialized
- 204 kAdeASEError
- 205 kAdeASIError
- 206 kAdeAsiNotInitialized
- 207 kAdeAsiConnectToEnvironmentFailed

Transactional Manager

1000	kErrClosed	Repeated attempts to close previously closed ADE object. Call support.
1001	kErrWasErased	Attempt to work with erased ADE object. Call support.

1002	kErrOpenForRead	Attempt to update ADE objects opened for read. Call support.
1003	kErrOpenForWrite	Attempts to get multiple access to ADE object opened for write. Call support.
1004	kErrWrongMode	Incorrect mode of the ADE objects should be opened. Call support.
1005	kErrClone	Exception at the time of cloning ADE objects. Call support.
1006	kErrResponse	Incorrect attempt to start ADE transaction. Call support.
1007	kErrObjIsAbsent	Attempt to work with erased ADE object. Call support.
1008	kErrAccess	Type of work with ADE does not correspond to its status. Call support.
1009	kErrMultipleUsage	Attempts to get multiple access to ADE object opened for write. Call support.
1010	kErrUpgrade	Incorrect attempt to update opening mode of the ADE object. Call support.
1011	kErrNotClosed	Object was not closed during current ADE transaction. Call support.
CAdeList		

- 1100 kAdeGetIteratorFails
- 1101 kAdeListIsEmpty

CAdeListIterator

1102 kErrListEnd

1103 kErrListObjectIsAbsent

Drawing

1200	kAdeQueriedEntity	Ignoring queried entity for saving selection. When using ADESAVEOBJS, the message means queried objects that are selected for save back are ignored.
1201	kAdeDwgNotActive	Attempt to read the object locked from an inactive drawing. When using ADEWHOHASIT, the message means the drawing from which the object was queried is no longer active. ADE is unable to determine if the object selected is currently locked. Activate the specified drawing and re-enter the command.
1202	kAdeReadDwgFileFails	ADE fails to read external drawing. If using ADEDRAWINGS, this message means that ADE is unable to read the specified drawing. Perhaps the drawing

		doesn't exit or the specified file is not a valid AutoCAD Drawing File. Check the error message stack for more information.
1203	kAdeMultipleUsers	ADE fails to access external drawing. If using ADEDRAWINGS, this message means ADE is unable to access the specified drawing because of file locking problems. Check the error message stack for more information.
1204	kAdeEntityLockingFails	ADE fails to lock. If using ADESELOBJS, this message means ADE is unable to lock an object. The object may already be locked. Use the ADEWHOHASIT to determine if the object is locked, perhaps by another user. Check the error message stack for more information.
1205	kAdeUnlockedEntity	Entity is unlocked. If using ADEWHOHASIT, this message indicates that

		the selected object is not currently locked.
1206	kAdeLockedEntity	Entity has been locked by another ADE user. If using ADESELOBJS, this message indicates that the object is already locked by another user. Use the ADEWHOHASIT command to identify the user.
1207	kAdeAlreadyInSaveSet	
1208	kAdeAlreadyNotInSaveSet	
1209	kAdeMultipleReaders	
1210	kAdeOpenPrefDictionaryFails	ADE fails to open ADE preferences dictionary. Call support.
1211	kAdeSavePrefDictionaryFails	ADE fails to save ADE preferences dictionary. Call support.
1212	kAdeRestoreDSetFails	ADE fails to restore the drawing set. Call support.
1213	kAdeOpenDSetDictionaryFails	ADE fails to open ADE drawing set dictionary. Call support.
1214	kAdeSaveDSetDictionaryFails	ADE fails to save ADE drawing set in

		dictionary. Call support.
1215	kAdeOpenQueryDictionaryFails	ADE fails to open ADE query library dictionary. Call support.
1216	kAdeSaveQueryDictionaryFails	ADE fails to save ADE query library in dictionary. Call support.
1217	kAdeOpenRTableDictionaryFails	ADE fails to open ADE range table dictionary. Call support.
1218	kAdeSaveRTableDictionaryFails	ADE fails to save ADE range table in dictionary. Call support.
1219	kAdeRestoreRTableDictionaryFails	ADE fails to restore ADE range table in dictionary. Call support.
1220	kAdeOpenDocViewDictionaryFails	ADE fails to open ADE Doc View information dictionary. Call support.
1221	kAdeSaveDocViewDictionaryFails	ADE fails to save ADE Doc View information in dictionary. Call support.

1222	kAdeOpenKeyViewDictionaryFails	ADE fails to open ADE Key View information dictionary
1223	kAdeSaveKeyViewDictionaryFails	ADE fails to save ADE Key View information in dictionary. Call support.
1224	kAdeSaveProjectionFails	ADE fails to save projection code in the drawing. Call support.
1225	kAdeCopyHardPointerFails	ADE fails to apply property alteration for some symbol table. Call support.
1226	kAdeDwgToBeReloaded	
1227	kAdeDwgHasBeenModified	There were objects queried from the drawing that will be treated as new objects. If using ADEDRAWINGS, this message means that when a drawing from which objects have been queried is detached, ADE converts the objects into newly created objects. When you use the ADEWHOHASIT command to see the origin of these objects,
		it says that they have not been queried.
------	-------------------	---
1228	kAdeOnLockedLayer	Objects from a locked layer have been selected. If using ADESELOBJS, ADEREMOBJS, or ADESAVEOBJS, this message means you selected objects from a locked layer and they cannot be added to, saved to, or removed from the save set or saved to source.
1229	kAdeDwgSaveFales	ADE fails to save the source drawing. If using ADESAVEOBJS, ADESELOBJS, ADEREMOBJS, OPEN, NEW, or QUIT, or if you're configuring options, or modifying objects, you may get this message. Call support. NOTE: Use this message exactly as spelled here.
1230	kAdeDwgLocksLeft	ADE fails to remove object locks (if present) by the end of ADE session. If using OPEN, NEW, or QUIT, or if you're configuring

		options, this message means the drawing might be locked by another ADE user.
1231	kAdeLinkWillBeLost	ADE does not save links between queried objects and source drawings between ADE sessions. Detach source drawing with queried objects; Use the SAVE command with queried objects or no objects in the save set.
1232	kAdeDwgDiskFull	
1233	kAdeDwgHasQueriedObject	
1234	kAdeDwgHasNoHeavyLock	
Current S	ession	
1300	kAdeGetCPointFails = 1300	ADE fails to get Text location for the entity. Call support.
1301	kAdeSetCPointFails	ADE fails to store Text location for the entity. Call support.
1302	kAdeTextInsPointMissed	Missing Text insert point. Call support.
1303	kAdeTextAllignPointMissed	Missing alignment point for Aligned and Fit Text.

Call support.

1304	kAdeTextHeightMissed	Missing Text height. Call support.
1305	kAdeTextStringMissed	Missing Text string. Call support.
1306	kAdeIndexUpdateFails	Exception in Index generation or regeneration. If using ADESAVEOBJS, ADESELOBJS, ADEREMOBJS, or ADEDWGMAINT, this message means entity modification occurred. Call support.
1307	kAdeIgnorePreview	ADE custom object is selected to add to save set or to save. If using ADESAVEOBJS, ADESELOBJS, or ADEREMOBJS this message has the following meaning: When doing a Preview Query, ADE creates a special object called a PREVIEW object, used to display the queried objects. PREVIEW cannot be saved back to a source drawing. When a user selects this object for adding to the save set or when saving back, ADE detects this and prevents

the operation.

1308	kAdeIgnorePSpace	
1309	kAdeEraseIRDObjectFails	ADE fails to erase the Object Data Table. Call support.
1310	kAdeRenameIRDObjectFails	ADE fails to rename the Object Data Table. Call support. If using ADEDEFDATA, this message means ADE can't rename Object Data Table. Perhaps Object Data of the same name already exists in the drawing or Object Data Table definitions bearing the old name are different in the source drawings. Check the error message for more information.
1311	kAdeAlterIRDObjectFails	ADE fails to alter Object Data Table. If using ADEDEFDATA, this message means ADE can't alter Object Data Table. Perhaps Object Data Table definitions are different in the source drawings. Check the error message stack for more information.
1312	kAdeIrdDuplicateTableName	Object Data Table with specified name already

		exists. ADEDEFDATA - all Object Data table names must be unique. Duplicate names are not allowed.
1313	kAdeIrdNotIdenticaFormat	Conflict in Object Data Table definition. If using ADEDEFDATA, this message means that perhaps Object Data Table definitions are different in the source drawings. NOTE: Use this message exactly as spelled here.
1314	kAdeBHatchUnit	ADE treats bhatch and its boundary as one unit. If using ADESELOBJS or ADEREMOBJS, this message has the following meaning: When adding an object to the save set, ADE checks if this object is part of a hatch boundary. If it is, all other objects that form the boundary are added to or removed from the save set. When saving back bhatched areas, ADE always treats the boundary as one object.
1315	kAdeReQuery	ADE queried one or more objects twice. ADE does not support UNDO for this operation. If using ADEQUERY or

ADERUNXQUERY this message has the following meaning: If a drawing file has been modified by another ADE user and if a queried object matches another query, ADE removes the old copy and queries a new copy. This operation can't be undone.

1316 kAdeCantAccessFont kAdeCantAccessImageFile 1317 1318 kAdeUnableRedefineXrefBlock **Drawing Set** kAdeRestoreDrawingSetFails ADE fails to read Drawing 1500 Set from DWG file. You encounter this message during ADE initialization or when using Open drawing file or ADEDRAWINGS. If using ADEDRAWINGS attach drawing, the message means "activate on attach" is ON or ade_dsattach(), ade_dswcattach() drawing is corrupted or old ADE version is in use. 1501 kAdeDuplicatedDrawing

Attempt to attach the same drawing twice. If using ADEDRAWINGS, this

		error appears when a drawing with the same name has already been attached to the work session.
1502	kAdeNestedDrawing	Prohibited attempt to edit properties of the drawing in the nested drawing set. If using ADEDRAWINGS, this message means a user is not allowed to modify the transformation and save back extents of nested drawings in the work session. These properties can only be modified for top level drawings.
1503	kAdeActivateDrawingFails	ADE fails to activate source drawing. ADE was unable to activate a drawing Perhaps the drawing does not exist or it is locked by another user, or the current user doesn't have permission to read the specified drawing file. Check the error message stack for more information.
1504	kAdeDeactivateDrawingFails	ADE fails to deactivate source drawing. ADE is unable to deactivate a drawing. Perhaps the

		drawing is locked by another user, the drawing no longer exists, or there are locked objects in the drawing. Check the error message stack for more information.
1505	kAdeLongDrawingDescription	Specified drawing description exceeds 133 symbols. Shorten description.
1506	kAdeEntityHasBeenLocked	Drawing with locked entities can't be deactivated. ADEDRAWINGS - the specified drawing cannot be deactivated because it contains locked objects. Remove locks and deactivate.
1507	kAdePreviewNotSupported	ADE 2.0 ignores these objects: Tolerance, Body, Ellipse, 3dSolid, Region, and Mline. If using ADEQUERY (Preview), ADEQVIEWDWGS, or ADEKEYVIEW, this message means ADE does not support a Preview Query of Tolerance, Body, Ellipse, 3dSolid, Region and Mline.

1509 kAdeActivateDrawingCancelled

1510 kAdeAttachDrawingCancelled

Feature Alteration

1800	kAdeInvalidFeatureType	Attempt to set an invalid property alteration type. If calling API functions ade_altpsetprop or ade_altpdefine, check for a mistake in property type.
1801	kAdeNoListId	Property alteration internal list is invalid. Call support.
1802	kAdeInvalidExpType	Attempt made to set an invalid property alteration expression type. Call support.
1803	kAdeNoExpression	Property alteration internal object is invalid. Call support.
1804	kAdeTextCreationFailed	Property alteration was unable to create a new text object. This message occurs when using queries that alter properties. Check expressions in the text property alteration definition.
1805	kAdeHatchCreationFailed	Property alteration was unable to create a new hatch object. This message occurs when using queries that alter properties. Check expressions in the hatch property alteration definition.
1806	kAdeInvalidColor	Invalid color passed to property

		alteration. This message occurs when using queries that alter properties. Check expressions evaluates to a valid AutoCAD color.
1807	kAdeInvalidLayer	Invalid layer name. This message occurs when using queries that alter properties. Check expressions evaluates to a valid AutoCAD layer.
1808	kAdeInvalidStyle	Invalid style name. This message occurs when using queries that alter properties. Check expressions evaluates to a valid style.
1809	kAdeInvalidJustification	The expression for justification in a text property alteration did not evaluate to a valid justification. This message occurs when using queries that alter properties. Check expressions evaluates to a valid AutoCAD justification.
1810	kAdeInvalidScale	
1811	kAdeNoRangeId	ADE internal object is invalid. Call support.
Mapping		
1900	kErrMapCoincPoint	Coincident points. If using ADERSHEET or ADETRANSFORM this message means either old or

		new points are coincident. They must be different.
1901	kErrMapWrongScale	Invalid scale. Call support.
1902	kErrMapTransform	Can't transform entity. ADERSHEET, ADETRANSFORM, ADEQUERY, ADESAVEOBJS An error appeared at the time of entity transformation. It is high-level error. There must be another error in the stack with more specific information.
1903	kErrMapWrongExtents	Invalid entity extents. If you're using ADETEXTLOC, ADERSHEET, ADEQUERY, ADESAVEOBJS, or calling AcDbEntity::getGeomExtents(method you may get this message.
1904	kErrMapWrongPoints	Invalid points number. If you're using ADERSHEET, this message means that the numbers of old and new points are different, or less than 2. Dialog doesn't allow this. C
1905	kErrMapWrongSelSet	Invalid selection set. Call support.
1906	kErrMapWrongEntityName	Invalid entity name. The entity is open. For example, it has been received from the API.
1907	kErrMapOpenEntity	Can't open entity. The entity is

		open. For example, another application opened the entity.
1908	kErrMapUpgradeEntity	Can't upgrade open. Entity modification occurred.
1909	kErrMapMoveStretchPoints	Can't modify stretch points. Call support.
1910	kErrMapEntityPoint	Can't modify entity points. Cal support.
1911	kErrMapCmdecho	Can't change CMDECHO variable. Using ADEFILLPOLYG may produce this message.
1912	kErrMapCecolor	Can't change CECOLOR variable. Using ADEFILLPOLYG may produce this message.
1913	kErrMapHatch	Error in hatch command. Usin ADEFILLPOLYG may produce this message.
1914	kErrMapWrongIntersectForPoints	Can't find intersection. Using ADEQUERY or calling AcDbentity::IntersectWith() method may produce this message.
1915	kErrMapWrongHandle	Wrong entity handle. Call support.
1916	kErrMapNotPolyline	Entity isn't polyline. Call support.

1917	kErrMapIterator	Can't create iterator. Call support.
1918	kErrMapWriteXData	Can't write Xdata. Ensure that Xdata size is 16 KB or less.
1919	kErrMapBuffer	Can't create buffer. If you're using ADEQUERY, to make a location query using a bufferfence, you may get this message.
1920	kErrMapStretchPoints	Can't get stretch points. If you're using ADETRANSFORM or ADEQUERY or calling AcDbEntity::getStretchPoints() you may get this message.
Topology		
2000	kErrTopInvalidName	Invalid topology name. Occurs during topology creation.
2001	kErrTopExist	Topology already exists. Occurs during topology creation.
2002	kErrTopBuildNet	Error building network topology. Occurs during topology creation.
2003	kErrTopBuildPolygon	Error building polygon topology. Occurs during topology creation.
2004	kErrTopBuildNode	Can't create node. Occurs

		during topology creation.
2005	kErrTopBuildArc	Can't create link. Occurs during topology creation.
2006	kErrTopBuildCntr	Can't create centroid. Occurs during topology creation.
2007	kErrTopAPIReg	Can't register topology API. Occurs during ADE loading.
2008	kErrTopFuncNotAvail	Function isn't available. Occurs if you're using topology functions of the API.
2009	kErrTopWriteData	Error writing Xdata. Occurs during topology creation and modification.
2010	kErrTopNotExist	Topology doesn't exist. Occurs if you're using topology functions of the API.
2011	kErrTopOverlayType	Wrong overlay type.
2012	kErrTopMakeLayer	Can't create new layer. Using ADEDWGCLEAN produces this message.
2013	kErrTopBlockNotExist	Block doesn't exist.
2014	kErrTopNotOpenForWrite	Topology isn't open for write. Occurs when editing topology.

2015	kErrTopOpenIrdTable	Can't open object data table. Occurs when loading and editing topology.
2016	kErrTopWrongIrdAttr	Invalid object data table. Occurs when loading and editing topology.
2017	kErrTopLoaded	Topology is already loaded. Occurs when loading topology.
2018	kErrTopIncompleteElem	Incomplete topological element. Occurs when editing topology.
2019	kErrTopInvalidColor	Invalid color number. Occurs when using ADEDWGCLEAN and creating topology.
2020	kErrTopInvalidFlag	Invalid flag. Occurs when using ADEDWGCLEAN.
2021	kErrTopInvalidTolerance	Invalid tolerance. Occurs when using ADEDWGCLEAN and creating topology.
2022	kErrTopInvalidCorridor	Invalid corridor width. Occurs when using ADEDWGCLEAN.
2023	kErrTopInvalidOffset	Invalid offset. Occurs when using buffering.

2024	kErrTopInvalidHeight	Invalid marker height. Occurs when using ADEDWGCLEAN.
2025	kErrTopInvalidMarkerType	Invalid marker type. Occurs when using ADEDWGCLEAN.
2026	kErrTopInvalidEntityType	Invalid type for new entities. Occurs when using ADEDWGCLEAN.
2027	kErrTopInvalidErrorType	Invalid error type. Occurs when using ADEDWGCLEAN.
2028	kErrTopIntersection	Intersections detected. Occurs when creating and editing polygon topology.
2029	kErrTopOverlayItself	Can't overlay topology with itself.
2030	kErrTopSourceDwgAccess	Can't access source drawing.
2031	kErrTopSourceDwgNotActive	Source drawing isn't active.
2032	kErrTopSourceDatabaseAccess	Can't access source drawing database.
2033	kErrTopSourceObjectId	Can't get object ID by handle in source drawing database.
2034	kErrTopNotLoaded	Topology isn't loaded.

2035	kErrTopImplicitNode	Node object doesn't exist in node topology.
2036	kErrTopMisplacedNode	Wrong node coordinates.
2037	kErrTopUnreferencedNode	Node isn't referenced in links.
2038	kErrTopUnexistentNode	Link references nonexistent node.
2039	kErrTopMismatchStartNode	Link has invalid ID at the start node.
2040	kErrTopMismatchEndNode	Link has invalid ID at the end node.
2041	kErrTopMisplacedCentroid	Wrong centroid coordinates.
2042	kErrTopMismatchLeftPoly	Link has invalid ID for the left polygon.
2043	kErrTopMismatchRightPoly	Link has invalid ID for the right polygon.
2044	kErrTopUnexistentCentroid	Centroid isn't inside polygon.
2045	kErrTopMultiplyCentroid	Polygon has several centroids inside.
2046	kErrTopWrongPolyQty	Some polygons are incorrect.
2047	kErrTopMismatchPolyArea	Incorrect polygon area.
2048	kErrTopMismatchPolyPerimeter	Incorrect polygon

perimeter.

2049	kErrTopOpenSourceDwgTopo	Topology loaded from source drawings can't be open for write.
2050	kErrTopOpenTempTopo	Temporary topology can't be open for write.
2051	kErrTopIdNotExist	Current drawing doesn't have OD table with information about last ID.
2052	kErrTopEmpty	Can't create or load empty topology.
2053	kErrTopWasModified	Topological objects were modified by AutoCAD commands.
2054	kErrTopMultiple	Object belongs to multiple topologies and can't be erased.
2055	kErrTopCalculateOffset	Can't calculate offset. Use default. Occurs when using buffering.
2056	kErrTopZeroOffset	Zero offset. Can't build buffer.
2057	kErrTopDifferentOffset	Offset has different sign for some objects. Can't build buffer.
2058	kErrTopInvalidSelSet	Invalid selection set. Occurs when using the

API.

2059	kErrTopCleanNotInit	Cleanup model isn't initialized. Occurs when using the API.
2060	kErrTopCleanNoGroup	There is no current group. Occurs when using the API.
2061	kErrTopCleanInvalidIndex	Invalid error index. Occurs when using the API.
2062	kErrTopCleanNoError	Current error isn't set. Occurs when using the API.
2063	kErrTopTraceLinkNotExist	Link doesn't exist in tracing model. Occurs when using the API.
2064	kErrTopTraceNodeNotExist	Node doesn't exist in tracing model. Occurs when using the API.
2065	kErrTopTraceNoPath	Result path isn't calculated. Occurs when using the API.
2066	kErrTopTraceInvalidIndex	Invalid element index. Occurs when using the API.
2067	kErrTopInvalidExpression	Can't process ADE expression. Occurs when using overlay, buffer, dissolve, or tracing command.

2068	kErrTopLockedTable	Can't write into topology OD table. Occurs when using dissolve command.
2069	kErrTopCreateTable	Can't create OD table. Occurs when using topology creation, overlay, buffer, or dissolve commands.
2070	kErrTopCreateTableColumn	Can't add column to OD table. Occurs when using topology creation, overlay, buffer, or dissolve commands.
2071	kErrTopTraceNodesEqual	Start and end nodes are the same. Occurs doing shortest path tracing.
2072	kErrTopTracePathNotExist	Empty path. Occurs during shortest path tracing.
2073	kErrTopTraceFloodNotExist	Empty path. Occurs when tracing floods.
2074	kErrTopRenameDisabled	Can't rename topology, because current drawing has queried objects with OD.
2075	kErrTopDeleteDisabled	Can't delete topology, because current drawing has queried objects with OD.

2076 kErrTopInvalidExtents

Topology API

2100	kAdeTopApiErrWro	ngInput	Missing or invalid parameter.
2101	kAdeTopApiWrongI	d	Invalid ID.
Tracing			
2150	kAdeTopSprErr	Tracing erro	or. Occurs during topology
Query Defin	ition		
2200	kErrUnexpectedBuff	Char	Invalid character encountered while reading the query definition from the drawing. Options are a) Recover the drawing b) Define and save a new query definition in the drawing.
2201	kErrInvalidIndex		An invalid line number was specified for either grouping or ungrouping of query lines. Specify the correct line number for grouping or ungrouping the lines.
2202	kErrInvalidQueryLir	10	One or more query lines have been incorrectly defined. May occur when you incorrectly place a parenthesis or an operator in a query line.

2203	kErrInvalidName	Either a query or a query category name is invalid. Ensure that the query or category name conform to the AutoCAd symbol name specifications.
2204	kErrEntryAlreadyExists	Either the query or the query category name already exists in the query library. Ensure that the query name is unique within the query library.
2205	kErrEntryInOtherCategory	The query name specified already exists in another category in the query library. Ensure that the query name is unique within the query library.
2206	kErrEntryAndFileAlreadyExist	The file name specified for saving the external query already exists. Choose a different file name.
2207	kErrASIConnectFailed	The connection to the ASI environment required for the SQL query was not made. Use ASE to connect to the environment before attempting to perform the SQL query.
2208	kErrASIStmtPrepareFailed	The call to CAsiExecStm:: Prepare failed. correct the table name or the SQL

		statement specified.
2209	kErrASICsrAllocFailed	The call to CAsiCsr::Allocate failed. Look at the ASI error displayed.
2210	kErrASICsrOpenFailed	The call to CAsiCsr::Open failed. Look at the ASI error displayed.
2211	kErrInvalidDOName	An invalid Environment, Schema or Catalog name was specified. Set the correct Environment, Catalog, and Schema names.
2212	kErrLPInitFailed	The call to CAseLinkPath::init failed. Look at the ASE error displayed.
2213	kErrColNotFound	Used for the SQL Order-by dialog now obsolete. Call support.
2214	kErrQDefNotInTM	The CAdeQueryDef object was not appended to the Transaction Manager. Internal error.
2215	kErrQryDefnExists	A query definition already exists and a new one cannot be loaded. Clear the existing query definition before loading a new one.

2216	kErrInvalidOperator	Invalid operator defined in query definition. The specified comparison operator is incompatible with operand types. Do not use > with the point type. Check the query definition and change either the operator or operand type.
2217	kErrInvalidPtrnOperator	Invalid operator defined in query definition for pattern value. If value operand is defined as pattern, only "=" comparison operator can be used. Check the query definition and change either operator or operand value.
2218	kErrInvalidField	Non-existent object data field specified. This error occurs when the user specifies the wrong object data field name for a table (if there is no such field in the specified table) in the query definition and executes the query. Check query definition and tables and correct the mistake.
2219	kErrInvalidNotBranch	
2220	kErrInvalidBranch	
2221	kErrUndefinedValue	

2222 kErrInvalidLocationType

2223 kErrCantLoadExternQuery

Query Manager

2300	kErrIntersectFailed		A call to CAseLinkSel:: intersectPartialKey failed. Look at the ASE error displayed.
2301	kErrNoTemplate		The query type was specified as report but no report options were defined. Define report options.
2302	kErrASIStoreValueFaile	ed	A call to CAsiData ::storeValue failed. Look at the ASI error displayed.
2303	kErrASIGetValueFailed	l	A call to CAsiData ::getValue failed. Look at the ASI error displayed.
Utility			
2400	kErrLicFatal	Fatal	error in ADE license. Call support.
2401	kErrFileNotFound	Can't when	find associated document. Occurs using ADEDOCVIEW.
2402	kErrPathNotFound	Can't using	find executable file. Occurs when ADEDOCVIEW.
2403	kErrBadFormat	Synta: Occur	x error in the command line. rs when using ADEDOCVIEW.
2404	kErrConvtErr	Error 2.0 da ADEC	converting ADE 1.0 data to ADE ta. Occurs when using CONVERT.

Data Dialogs

- 2450 kErrIRDMismatch
- 2451 kErrIRDInvalidName
- 2452 kErrIRDTableExists
- 2453 kErrInvalidTableName
- 2454 kErrInvalidAttrName
- 2455 kErrTopoName
- 2456 kErrQueriedAndNotNew
- 2457 kErrNotAdministrator

GenLink

2500 kErrTagNotF	ound
------------------	------

- 2501 kErrTagValueAbcent
- 2502 kErrIllegalFormat
- 2503 kErrColMoreThanOne

Environment

2600	kErrInitEnv	An error occurred during the
		initialization of ADE. The cause
		of this error may be due to errors
		in loading/initializing ADE user
		preferences, system preferences,
		log file, or user list.

2601 kErrCantFindAdeExePath

2602	kErrINIWrite		
2603	kErrInvalidUserNam	e	The user name specified does not exist in the user list. Use a user name that already exists in the user list or define a new one using User Administration.
2604	kErrLoadUserList		
2605	kErrSaveUserList		
2606	kErrInvalidPswd		The password specified does not match the one specified in the user list for this user. Use the correct password.
Rx			
2700	kErrRxAseLoad	ASE isn'	t loaded Can't initialize ASE API.
2701	kErrRxAseInit	Object D	ata module isn't loaded.
2702	kErrRxIrdLoad	Can't init	tialize Object Data API.
2703	kErrRxIrdInit	Specified found in	l coordinate system category not the library. Call support.
Projection			
2800	kErrNoProjCatFound	l	Specified coordinate system category not found in the library.Call support.
2801	kErrNoDatumFound		Specified coordinate system datum not found. Call support.
2802	kErrNoElipFound		Ellipsoid not found in the

		ellipsoid list. Call support.
2803	kErrNoCoordFound	Specified Coordinate System not found. Call support.
2804	kErrFaileOpenDatumFile	Can't open projection .mp3 file. Occurs when loading ADE.
2805	kErrFaileOpenElipFile	Can't open ellipse file. Occurs when loading ADE.
2806	kErrNoneCoord	Internal code to set "None" projection to the drawing. Call support.
Expression		
2900	kErrNoExpressionFound	Empty expression is specified. Call support.
2901	kErrGetPropFail	This is an internal code to show that entity has no specified property. Call support.
2902	kErrExpEvalFail	ADE fails to evaluate expression. Occurs when executing query with property alteration and executing a property query.
2903	kErrExpMissingQuote	Quotes mismatched in SQL expression. Occurs when executing a SQL query and a query with SQL property alteration. Also occurs when using ADECONVERT.
2904	kErrExpMissingCParen	Parenthesis mismatched. Occurs

		when executing a query with feature alteration and executing a property query.
2905	kErrExpExceedThreeOper	More than three operands are specified. Occurs when executing a query with property alteration and executing a property query.
2906	kErrRngTabNameExist	Range table with specified name already exists. Call support.
2907	kErrLpnInvalid	
2908	kErrLpnNotFound	
2909	kErrRangeInvalidElse	
For more	Expression errors, which begin	nat−15 click 🛛
Index		
Index 3000	kErrInvalidIndexVersion	The version of the index in the drawing is invalid. The options are: 1)Regenerate the index using drawing maintenance 2)Remove the index using the index removal utility and then re-generate the index.
Index 3000 3001	kErrInvalidIndexVersion kErrIndexOutOfDate	The version of the index in the drawing is invalid. The options are: 1)Regenerate the index using drawing maintenance 2)Remove the index using the index removal utility and then re-generate the index. The index in the drawing is out-of- date. Regenerate the index using drawing maintenance.
Index 3000 3001 3002	kErrInvalidIndexVersion kErrIndexOutOfDate kErrTypeAllObjects	The version of the index in the drawing is invalid. The options are: 1)Regenerate the index using drawing maintenance 2)Remove the index using the index removal utility and then re-generate the index. The index in the drawing is out-of-date. Regenerate the index using drawing maintenance.

Validation

3100	kErrWrongSymbolName
3101	kErrWrongSymbol
3102	kErrWrongStrLength
3103	kErrDirDoesNotExist
3104	kErrDirReadOnly
3105	kErrAccessDenied
3106	kErrFileDoesNotExist
3107	kErrFileAlreadyExists
3108	kErrFileOpenFailed
3109	kErrFileReadOnly
3110	kErrInvalidString
3111	kErrOutOfRange
3112	kErrWrongColor

3113 kErrIncorrectParameters One of ADE validation methods recognized incorrect input parameters. This error is an internal ADE error.

- 3114 kErrFileOpenLimit
- 3115 kErrShareViolation
- 3116 kErrNetAccessDenied

3117 kErrPathDoesNotExist

File Locking

3200	kErrDwkFileDoesNotExist	ADE lock file is locked. Occurs when using ADEDRAWINGS and ADEQUERY commands and when ADE is running in a multi-user environment.
3201	kErrOpenDwkFileFailed	ADE was unable to open the .DWK lock file. Call support.
3202	kErrFileLockedByAcad	Attempt to remove a user who does not exist from the lock file. Call support.
3203	kErrOldMapLockFile	ADE was unable to create the .DWK lock file. Call support.
3204	kErrFileIsNotDwk	ADE tried to lock a file for write that was already locked for read. Call support.
3205	kErrSpecifiedUserDoesNotExist	ADE tried to lock a file for read that was already locked for write. Occurs during query operations in a multi-user environment.
3206	kErrCreateDwkFileFailed	ADE tried to open and read a file that was not a valid .DWK file. Call

support.

3207	kErrFileIsLockedForRead	ADE was unable to unlock the lock file. Call support.
3208	kErrFileIsLockedForWrite	ADE tried to attach a file that is already open by AutoCAD. Occurs when using ADEDRAWINGS with ATTACH operations if the file is open in an AutoCAD project.
3209	kErrInvalidLockStateSpecified	ADE internal object is invalid. Call support.
3210	kErrNotOwnerOfWLH	ADE tried to remove a write lock when the user did not have a write lock. Call support.
3211	kErrUserIsNotWriter	
3212	kErrUserIsNotReader	
3213	kErrUserHasReadLock	
3214	kErrLockFileIsFull	
3215	kErrDwgFileDoesNotExist	ADE tried to unlock a file but the .DWK file was missing. Occurs if the .dwk file was erased after a file was attached.
3216	kErrNotAnADELockFile	ADE internal object is invalid. Call support.

3217	kErrFileMayHaveBeenModified	Existing .DWK file does not belong to ADE. the lock file exists and can be read by ADE, but ADE does not own the file.
3218	kErrFileHasLocks	ADE tried to remove a lock file but it was not found. Occurs if the .dwk file was erased after a file was attached.

Unicode Support

- 4000 kAdeUnicodeInsufficientBufferToConvert
- 4001 kAdeUnicodeInvalidFlagsToConvert
- 4002 kAdeUnicodeInvalidParameterToConvert
- 4003 kAdeUnicodeNoTranslation
- 4004 kAdeUnicodeCodePageNotAvailable

Double-Byte Support

4005 kAdeNoMBCSAllowed

Macros

An AutoCAD Map VBA macro is a program that interacts with AutoCAD Map through the Automation API. The VBA macro may include operations on projects, drawings, drawing sets, queries, and object data. VBA macros are stored in .dwg or .dvp format. You create macros in modules, the windows in the Visual Basic Editor for entering VBA code.

From AutoCAD Map, you can create macros or access macros in all active drawings and projects, or just in certain ones. Click Tools > Macros > Macro. Click VBA Manager to load, unload, save, create, embed, and extract VBA projects. For more information, click Help in the Macros dialog box.

What else do you want to do?

Get started with VBA

Use modules

Create a simple macro

Save macros

Work with projects

<u>Use the Object Browser</u>

Insert files into modules

Find Help on Microsoft VB and VBA

Find Help on AutoCAD APIs

You can also use the browse buttons to navigate this topic group.

/BA macro is a program that interacts with AutoCAD Map tion API. The VBA macro may include operations on rawing sets, queries, and object data. VBA macros are p format. You create macros in modules, the windows in the for entering VBA code.

p, you can create macros or access macros in all active ts, or just in certain ones. Click Tools > Macros > Macro. to load, unload, save, create, embed, and extract VBA formation, click Help in the Macros dialog box. do?

4

<u>0</u>1

<u>'ser</u>

<u>ıles</u>

oft VB and VBA

AD APIs

browse buttons to navigate this topic group.
Comparing VBA and VB

VBA and VB have different entry points to the object model. The VB entry point is the application, whereas with VBA you are already in the application. Also, with VB you have to reference the AcadApplication.ActiveDocument property in the AutoCAD Automation API explicitly, but with VBA you can use the global "alias" for the ActiveDocument property, ThisDrawing.

For information about writing VB versus VBA code, click **N**.

BA and VB

ifferent entry points to the object model. The VB entry point nereas with VBA you are already in the application. Also, reference the AcadApplication.ActiveDocument property comation API explicitly, but with VBA you can use the ActiveDocument property, ThisDrawing.

Managing drawing sets

DrawingSet collections contain the attached drawings of projects. You attach drawings to a project by calling the DrawingSet.Add function, as follows:

Dim atdr As AttachedDrawing Dim amap As AcadMap

```
Set amap = ThisDrawing.Application. _
GetInterfaceObject("AutoCADMap.Application")
Set atdr = map.Projects(ThisDrawing).DrawingSet.Add("C:\\campus.dwg")
```

Using the GetTableList function of the DrawingSet, you can access AutoCAD symbol tables. You can get names of layers, line types, blocks, object data tables, and other symbol tables in a drawing.

The DrawingSet provides the frequently-used zoom extents feature.

boolVal = map.Projects(ThisDrawing).DrawingSet.ZoomExtents

The DrawingSet.Extents method gets extents of the drawing, which can define a query window, as shown in the following example.

Dim wind As WindowBound Dim dblary As Variant

```
'Get DWG Extents
dblary = prj.DrawingSet.Item("MAPTUT:\\citymap7.dwg").Extents
```

'Define Boundary Area for Location Set mapu = map.Projects(ThisDrawing).MapUtil Set wind = mapu.NewWindow(_ mapu.NewPoint3d(dblary(0), dblary(1), 0), _ mapu.NewPoint3d(dblary(2), dblary(3), 0))

Through the AttachedDrawing objects in the DrawingSet collection, you activate and nest drawings, transform and save back drawing objects, and get information on a drawing, such as its alias or locked status.

awing sets

ons contain the attached drawings of projects. You attach t by calling the DrawingSet.Add function, as follows:

dDrawing ⁄Iap

wing.Application. _
'AutoCADMap.Application")
cts(ThisDrawing).DrawingSet.Add("C:\\campus.dwg")

ist function of the DrawingSet, you can access AutoCAD an get names of layers, line types, blocks, object data tables, oles in a drawing.

vides the frequently-used zoom extents feature.

```
cts(ThisDrawing).DrawingSet.ZoomExtents
```

ents method gets extents of the drawing, which can define a own in the following example.

wBound

nt

sSet.Item("MAPTUT:\\citymap7.dwg").Extents

rea for Location jects(ThisDrawing).MapUtil wWindow(_ blary(0), dblary(1), 0), _ blary(2), dblary(3), 0))

dDrawing objects in the DrawingSet collection, you activate ansform and save back drawing objects, and get information s its alias or locked status.

Queries

Query object

A query is composed of query branch and query leaf objects. Each leaf defines a single query condition, and each branch defines a condition grouping. A query has a main branch and one or more leaves, and may contain additional subbranches and leaves. Branch and leaf objects are related by properties that specify their logical operators. The following figure illustrates this structure for a simple query.



The query selects drawing objects inside a window boundary, and that are either on the STREAM layer or bodies of water with an average depth of less than 10 feet. For step-by-step information about creating queries, click **B**.

After defining a query, if you change leaves using the QueryLeaf.Value property, call Query.Define again before executing the query. For step-by-step information about modifying queries, click **2**.

l of query branch and query leaf objects. Each leaf defines a n, and each branch defines a condition grouping. A query id one or more leaves, and may contain additional sub-Branch and leaf objects are related by properties that operators. The following figure illustrates this structure for a

<location< th=""><th>Inside Window></th><th></th></location<>		Inside Window>	
R	Leaf:	<property layer='``STREAM"'></property>	
R 🔸	Leaf:	<objectdata 10;<="" <="" td="" water_bodies.avg_depth=""><td>></td></objectdata>	>

awing objects inside a window boundary, and that are either er or bodies of water with an average depth of less than 10 > information about creating queries, click ■.

cy, if you change leaves using the QueryLeaf.Value property, ain before executing the query. For step-by-step information ries, click ■.

Altering queried objects

If property alteration is defined and enabled for a query, certain properties of queried objects are altered when the query result is displayed in the project drawing. (There is no effect on source drawings unless you deliberately save the objects back.) Property alteration is governed by a query's property alteration definition, an Alterlines collection, which can contain any number of simple alterations (AlterLine objects), hatch alterations (HatchDef objects), or text alterations (TextDef objects). To alter queried objects, first define AlterLine, HatchDef, or TextDef objects, and then add them to the AlterLines collection of the current query. For step-by-step information about altering queried objects,

Conditional property alterations

Note that you can specify conditional property alterations—for example, to color a queried object green if its width is in the range 0 to 1, or color it blue otherwise. First define a range table by building a RangeTable collection, and then use a range table expression instead of an explicit value or expression when you set the Value parameter of a property alteration (or the TextValue parameter of a text alteration, or the Pattern property of a Hatch alteration). For step-bystep information about using range tables, click .

ied objects

 $\frac{ee Also}{1s}$ defined and enabled for a query, certain properties of ltered when the query result is displayed in the project construction of the second sec rty alteration is governed by a query's property alteration nes collection, which can contain any number of simple e objects), hatch alterations (HatchDef objects), or text objects). To alter queried objects, first define AlterLine, of objects, and then add them to the AlterLines collection of r step-by-step information about altering queried objects,

rations

ecify conditional property alterations—for example, to color in if its width is in the range 0 to 1, or color it blue ie a range table by building a RangeTable collection, and e expression instead of an explicit value or expression when ameter of a property alteration (or the TextValue parameter r the Pattern property of a Hatch alteration). For step-byut using range tables, click .

Object data

Object data tables store data that you attach to objects. The

MapUtil.NewODFieldsDef function creates the structure of an object data table. The ODTables.Add method applies that structure to a specified table, registers it in the project, and specifies the type of object data, either extended entity data or an Xrecord. The Value properties of ODFieldDef objects, which are members of the ODRecord collection, represent the actual object data. The ODRecord object attaches data to objects. The collection of records contains data for an object that may be defined in different tables.

For step-by-step information about creating object data tables, click 🗷

ore data that you attach to objects. The ldsDef function creates the structure of an object data table.

method applies that structure to a specified table, registers it ecifies the type of object data, either extended entity data or ue properties of ODFieldDef objects, which are members of ction, represent the actual object data. The ODRecord object cts. The collection of records contains data for an object that fferent tables.

rmation about creating object data tables, click 🗷

Error handling

See Also

The AutoCAD Map Automation API provides objects for error handling in addition to the error handling objects available through VBA. When an error occurs in the execution of an AutoCAD Map Automation API function, the API pushes errors onto the error stack. Using the ErrorStack object, you can get a diagnostic message of the error and display it for the user. To better interpret error messages or to trigger the appropriate error handler, you can determine the following types of errors by reading the ErrType property of the error in the error stack: ADE, ASE, AutoCAD, ASI, object data executable (IRD), coordinate system library (mentor), Windows application, and user-defined (DiagMessage) errors.

The DiagParam object represents user-defined error messages. By calling ErrorEntry.Add, you can add such a message to the error stack. You add your own messages to the Error Stack in conjunction with defining a failure condition for your macro.

For step-by-step information about creating an error handler, click 🔳

Automation API provides objects for error handling in handling objects available through VBA. When an error on of an AutoCAD Map Automation API function, the API le error stack. Using the ErrorStack object, you can get a of the error and display it for the user. To better interpret trigger the appropriate error handler, you can determine the rors by reading the ErrType property of the error in the error utoCAD, ASI, object data executable (IRD), coordinate or), Windows application, and user-defined (DiagMessage)

ct represents user-defined error messages. By calling can add such a message to the error stack. You add your Error Stack in conjunction with defining a failure condition

rmation about creating an error handler, click 🔳

EAlterationType

Property alteration types

00	kAlterationBlockName	Block name
01	kAlterationColor	Color
02	kAlterationLayer	Layer name
03	kAlterationRotation	Rotation
04	kAlterationElevation	Z coordinate in the user coordinate system
05	kAlterationHeight	Text height
06	kAlterationLineType	Line type
07	kAlterationScale	Scaling factor. For example "1.2" = 120%
08	kAlterationStyle	Text style
09	kAlterationText	Text value
10	kAlterationThickness	Thickness
11	kAlterationWidth	Line width
12	kAlterationTextEntity	Text entity definition
13	kAlterationHatch	Hatch definition

ype

/pes

3lockName	Block name
Color	Color
Jayer	Layer name
Rotation	Rotation
llevation	Z coordinate in the user coordinate system
leight	Text height
lineType	Line type
cale	Scaling factor. For example "1.2" = 120%
style	Text style
lext	Text value
hickness	Thickness
Vidth	Line width
extEntity	Text entity definition
Iatch	Hatch definition

EPropertyType

Property condition types

00	kArea
01	kBlockName
02	kColor
03	kElevation
04	kEntType
05	kGroup
06	kLayer
07	kLength
08	kLineType
09	kTextStyle
10	kTextValue
11	kThickness
12	kFeature
13	kLineweight
14	kPlotstyle

)e

/pes

е

:

QueryLeaf object

Represents a query Condition. See Also

Contained by a query branch, which is contained by a query. Created by adding a leaf to a query using QueryBranch.Add. For more information, click .

oject

<u>)bject Model</u> <u>See Also</u> ondition.

y branch, which is contained by a query. leaf to a query using QueryBranch.Add. n, click ■.

AlterLine object

Methods <u>Properties</u> <u>Object Model</u> <u>See Also</u> A property alteration.

Created by calling AlterLines.Add.

Contained by the property alteration definition of a query, an AlterLines collection.

AlterLine objects represent the simple property alterations; for example, a change of color. Text and hatch alterations, which add text labels and apply hatch patterns to queried objects, are represented by two additional objects, TextDef and HatchDef. A property alteration definition can contain any of the three.

For more information, click .

ect

Object Model See Also

lterLines.Add.

perty alteration definition of a query, an AlterLines

present the simple property alterations; for example, a t and hatch alterations, which add text labels and apply hatch bjects, are represented by two additional objects, TextDef perty alteration definition can contain any of the three.

n, click 🛎.

HatchDef object

Methods <u>Properties</u> <u>Object Model</u> <u>See Also</u> A hatch alteration. Hatch alterations add hatch patterns to queried objects if they are closed figures; for example, closed polylines and circles.

Created by AlterLines.Add or AlterLines.AddHatchDef.

Contained by the property alteration definition of a query, an AlterLines collection.

For more information, click .

ect

<u>Object Model</u> <u>See Also</u> atch alterations add hatch patterns to queried objects if they r example, closed polylines and circles.

es.Add or AlterLines.AddHatchDef.

perty alteration definition of a query, an AlterLines

n, click 🛎.

TextDef object

Methods <u>Properties</u> <u>Object Model</u> <u>See Also</u> A text alteration. Text alterations add text labels to queried objects.

Created by AlterLines.Add or AlterLines.AddTextDef.

Contained by the property alteration definition of a query, an AlterLines collection.

For more information, click .

Ct

<u>Object Model</u> <u>See Also</u> tt alterations add text labels to queried objects.

es.Add or AlterLines.AddTextDef.

perty alteration definition of a query, an AlterLines

n, click 🛎.
ProjectOptions object

Methods <u>Properties</u> <u>Object Model</u> <u>See Also</u> Represents options for individual AutoCAD Map projects.

Contained by Project objects.

Created automatically when a project is added to the Projects collection.

Options for the AutoCAD Map application are represented by the SystemOptions object. For more information about options, go to "Using AutoCAD Map > Setting Options > Setting System Options" in AutoCAD Map UI Help.

ProjectOptions Properties

The ProjectOptions object has no methods. Its properties are listed in three groups below:

Project Options Query Options Saveback Options

Project Options

ActivateDwgsOnAttach property	Boolean
AdjustSizesAndScalesForChangesInUnits property	Boolean
AdjustRotationsForMapDistortions property	Boolean
AdjustSizesAndScalesForMapDistortions property	Boolean
AdjustElevations property	Boolean
AdjustZeroRotationObjects property	Boolean
NoOfSQLConditionsInHistory property	Long
RestoreLastActiveOnStartup property	Boolean
ReconnectDbOnWSOpen property	Boolean

Query Options

CaseSensitiveMatch property	Boolean
SaveCurrQueryInSession property	Boolean
MkSelSetWithQryObj property	Boolean
DefaultJoinOperator property	EJoinOperator
ColorForAdd property	String
ColorForRemove property	String
BlockLocnForQuery property	Boolean
TextLocnForQuery property	Boolean
ReferenceBoundaryForAreaLocation property	Boolean
ShowBlockAsInsPt property	Boolean
CreateAssociativeHatch property	Boolean
ShowPreviewImageAsBoundaryOnly property	Boolean

Saveback options

DontAddObjectsToSaveSet property	Boolean
MarkObjectsForEditingWithoutPrompting property	Boolean
RedefineBlockDefinitions property	Boolean
RedefineLayerDefinitions property	Boolean
RedefineTextStyleDefinitions property	Boolean

RemoveUnusedGroups property	Boolean
EraseSavedBackObjects property	Boolean
RemoveLockAfterSave property	Boolean
CreateHistoryFileOfChanges property	Boolean
CreateBackupFileOfSourceDwg property	Boolean

AcadMap object

Methods <u>Properties</u> <u>Object Model</u> <u>See Also</u> Top-level object for accessing other objects.

Contains drives aliases, projects, and error stack collections and the system options object.

Created by getting an AutoCAD Automation interface object.

AutoCAD Map Automation and AutoCAD Automation are integrated APIs. Using AutoCAD Automation, you get an AcadMap object. AutoCAD Map Automation organizes objects in a tree-like fashion with AcadMap at the top level. AcadMap contains lower-level objects, and these objects contain even lower-level objects. You navigate the hierarchy of contained objects to access AutoCAD Map functionality. For example, from the contained Projects, you get a Project, and access lower levels of functionality through the query, drawing set, object data tables, project options, and other objects.

The following code creates an AcadMap object and gets the project associated with the active drawing.

Dim amap As AcadMap Dim prj As Project

Set amap = ThisDrawing.Application. _ GetInterfaceObject("AutoCADMap.Application") Set prj = amap.Projects(ThisDrawing)

ect

<u>Object Model</u> <u>See Also</u> accessing other objects.

es, projects, and error stack collections and the system

1 AutoCAD Automation interface object.

omation and AutoCAD Automation are integrated APIs. tomation, you get an AcadMap object. AutoCAD Map es objects in a tree-like fashion with AcadMap at the top ains lower-level objects, and these objects contain even (ou navigate the hierarchy of contained objects to access tionality. For example, from the contained Projects, you get lower levels of functionality through the query, drawing set, oject options, and other objects.

creates an AcadMap object and gets the project associated ing.

Лар

ving.Application. _
'AutoCADMap.Application")
cts(ThisDrawing)

Alias object

Methods <u>Properties</u> <u>Object Model</u> <u>See Also</u> Represents a shorthand name for a drive and directory.

Properties are a full path that exists on the hard drive and the name of the alias.

Created by calling Aliases.Add.

The following example uses both properties of this object to cycle through and report on the aliases in a project.

Dim als As Alias Dim cAls As Integer, i As Integer Dim amap As AcadMap Dim strOutput As String

Set amap = ThisDrawing.Application. _ GetInterfaceObject("AutoCADMap.Application") cAls = amap.Aliases.Count For i = 0 To cAls - 1 Set als = amap.Aliases.Item(i) strOutput = strOutput & als.Name & "=" & als.Directory & Chr(13) Next i MsgBox strOutput <u>Object Model</u> <u>See Also</u> nd name for a drive and directory.

path that exists on the hard drive and the name of the alias.

liases.Add.

ple uses both properties of this object to cycle through and in a project.

', i As Integer ∕Iap ring

```
ving.Application. _
'AutoCADMap.Application'')
.Count
```

```
es.Item(i)
It & als.Name & "=" & als.Directory & Chr(13)
```

Aliases collection

A collection of shorthand names of drives and directory locations.

Contained by an AcadMap object.

Created by calling Aliases.Add.

This collection has the typical methods of a collection object: Add, Item, and Remove plus additional capabilities for adding, modifying, deleting, and listing drive aliases. The FindByPath method gets an alias that corresponds to an actual path.

ction

<u>bject Model</u> <u>See Also</u> hand names of drives and directory locations.

ıdMap object.

liases.Add.

he typical methods of a collection object: Add, Item, and nal capabilities for adding, modifying, deleting, and listing ndByPath method gets an alias that corresponds to an actual

AllBound object

Methods Properties <u>Object Model</u> <u>See Also</u> Unlimited boundary.

Created by the MapUtil.NewLocationAll method.

This object defines an unlimited location condition, which returns all objects from all attached and active source drawings. Note that you can combine it with a *And Not xx* condition to retur *all objects but xx*. For example, combined with an *And Not Color = Red* property condition, a LocationAll condition returns all objects but the red ones.

The following example passes the AllBound object in the call to SetLocationCondition to query all objects in the attached drawings.

Dim amap As AcadMap Dim prj As Project Dim qry As Query Dim qrybr As QueryBranch Dim qrylf As QueryLeaf

```
Set amap = ThisDrawing.Application. _
GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
Set qry = prj.CurrQuery
Set qrybr = qry.QueryBranch
Set qrylf = qrybr.Add(kLocationCondition, kOperatorAnd)
qrylf.SetLocationCond kLocationInside, prj.MapUtil.NewLocationAll
```

ect

es <u>Object Model</u> <u>See Also</u>

Jtil.NewLocationAll method.

n unlimited location condition, which returns all objects l active source drawings. Note that you can combine it with on to retur *all objects but xx*. For example, combined with *Red* property condition, a LocationAll condition returns all nes.

ple passes the AllBound object in the call to n to query all objects in the attached drawings.

Лар

Branch Leaf

```
ving.Application. _
'AutoCADMap.Application")
cts(ThisDrawing)
ery
yBranch
d(kLocationCondition, kOperatorAnd)
nd kLocationInside, prj.MapUtil.NewLocationAll
```

AlterLines collection

A property alteration definition: <u>See Also</u> A property, hatch, and text alterations.

Contained by a query.

Created automatically by the project.

Used with a range table that modifies objects in different ways based on where they fall in a range of values. You can modify 14 properties, including text and hatch patterns, of queried objects using this collection. You add property, hatch, or text alterations to the collection using the Add, AddHatchDef, or AddTextDef methods. Therefore, to get an item from this collection, declare the return value as Variant. Source drawings are unaffected by property alterations unless you save them back.

To alter properties, you define a query, set the Query.AlterTag property to True, create a property alteration using Query.AlterProp, and execute the query in Draw mode. You cannot use property alteration in other modes.

For more information, click **N**.

llection

<u>Diject Model</u> See Also definition: a collection of property, hatch, and text

y.

y by the project.

ble that modifies objects in different ways based on where f values. You can modify 14 properties, including text and eried objects using this collection. You add property, hatch, the collection using the Add, AddHatchDef, or AddTextDef to get an item from this collection, declare the return value 'awings are unaffected by property alterations unless you

ou define a query, set the Query.AlterTag property to True, eration using Query.AlterProp, and execute the query in not use property alteration in other modes.

n, click 🛎.

AttachedDrawing object

An AttachedDrawing object represents a drawing attached to the project.

An AttachedDrawing object is contained by a DrawingSet collection, which is contained by a project.

An attached drawing can contain a NestedDrawings collection, or the drawing can be a member of another attached drawing's NestedDrawings collection.

An attached drawing can be active, inactive, or locked for querying. The objectlocking feature is necessary when users share a set of source drawings in a networking environment.

To use an AttachedDrawing object, you create a project containing drawings. You get the DrawingSet collection using the Project.DrawingSet property, and add an AttachedDrawing object to the DrawingSet collection using the DrawingSet.Add method. With access to an AttachedDrawing object, you manipulate your drawings using the AttachedDrawing object properties and methods. You can also manipulate a NestedDrawings collection using these properties and methods.

wing object

<u>bject Model</u> <u>See Also</u> 3 object represents a drawing attached to the project.

g object is contained by a DrawingSet collection, which is ct.

can contain a NestedDrawings collection, or the drawing another attached drawing's NestedDrawings collection.

can be active, inactive, or locked for querying. The objectcessary when users share a set of source drawings in a nent.

rawing object, you create a project containing drawings. Set collection using the Project.DrawingSet property, and ving object to the DrawingSet collection using the thod. With access to an AttachedDrawing object, you wings using the AttachedDrawing object properties and so manipulate a NestedDrawings collection using these)ds.

BufferFenceBound collection

A BufferFenceBound collection is a polygon boundary, also called a fence boundary, with a buffer zone around it.

Created with MapUtil.NewBufferFence method.

A buffer fence boundary is made of Point3d objects.

Bound collection

<u>Diject Model</u> <u>See Also</u> Collection is a polygon boundary, also called a fence fer zone around it.

il.NewBufferFence method.

lary is made of Point3d objects.

BufferPolylineBound collection

A polyline boundary with a buffer zone.

Created by the MapUtil.NewBufferPolyline method.

eBound collection

<u>bject Model</u> <u>See Also</u> with a buffer zone.

Jtil.NewBufferPolyline method.

CircleBound object

Methods <u>Properties</u> <u>Object Model</u> <u>See Also</u> A circle boundary.

Created by the MapUtil.NewCircle method.

The following example defines an area using a CircleBound object. First the code prompts the user to select the centre of the circle, and then to specify the radius. Lastly, it creates the boundary.

Dim acadapp As AcadApplication Dim amap As AcadMap Dim prj As Project Dim cir As CircleBound Dim acadu As AcadUtility Dim varVal As Variant, Radius As Double, boolVal As Boolean Dim strVal As String, ReturnValue As String

Set acadapp = ThisDrawing.Application Set amap = acadapp.GetInterfaceObject("AutoCADMap.Application") Set prj = amap.Projects(ThisDrawing) Set acadu = acadapp.ActiveDocument.Utility varVal = acadu.GetPoint(, "Select centre of circle: ") Radius = acadu.GetDistance(varVal, "Drag a line for radius: ") Set cir = prj.MapUtil.NewCircle(Radius, varVal(0), varVal(1))

object

Object Model See Also

Jtil.NewCircle method.

ple defines an area using a CircleBound object. First the er to select the centre of the circle, and then to specify the ites the boundary.

adApplication

und Utility Int, Radius As Double, boolVal As Boolean ;, ReturnValue As String

'rawing.Application GetInterfaceObject("AutoCADMap.Application") cts(ThisDrawing) .ActiveDocument.Utility oint(, "Select centre of circle: ") Distance(varVal, "Drag a line for radius: ") l.NewCircle(Radius, varVal(0), varVal(1))

ClosedPolylineBound collection

A closed polyline boundary. <u>See Also</u>

Created by the MapUtil.NewClosedPolyline method.

neBound collection

<u>)bject Model</u> <u>See Also</u> undary.

Jtil.NewClosedPolyline method.

DiagParam object

Methods <u>Properties</u> <u>Object Model</u> <u>See Also</u> Provides diagnostic information about an error.

Created by calling Error.Add.

A member of the ErrorEntry collection.

Properties include the ID of the object that caused the error, an error description, and if SQL processing was involved, the SQL statement, and location of the error within the statement.

To use this object, determine if the last error code is a kMapError. If so, retrieve the errors from the error stack to obtain diagnostic information represented by DiagParam.
bject

<u>Object Model</u> <u>See Also</u> information about an error.

rror.Add.

orEntry collection.

e ID of the object that caused the error, an error description, ig was involved, the SQL statement, and location of the ment.

etermine if the last error code is a kMapError. If so, retrieve rror stack to obtain diagnostic information represented by

DrawingSet collection

Represents a group of drawings attached to the project.

Contained by the Project object, a member of the Projects collection.

To use a DrawingSet collection, you create an AcadMap object and access the Projects collection using the AcadMap.Projects property. You then access the drawing set of a project using the Project.DrawingSet property. Once you have access to the DrawingSet collection, you use its methods to add or remove drawings from the set, get the drawings symbol tables, or display the drawing at different zoom settings.

ollection

<u>bject Model</u> <u>See Also</u> of drawings attached to the project.

ject object, a member of the Projects collection.

collection, you create an AcadMap object and access the sing the AcadMap.Projects property. You then access the ect using the Project.DrawingSet property. Once you have gSet collection, you use its methods to add or remove et, get the drawings symbol tables, or display the drawing at gs.

ErrorEntry collection

A collection of diagnostic parameters, one set for each error.

User-created by adding an error to the error stack, or API-created by an error condition.

Diagnostic parameters, such as the error message, provide clues about why the error occurred. Properties of the error entry include code, type, and message for the error. Methods are typical for a collection. For more information, click **2**.

The following example adds a nonexistent file to the drawing set and shows how to display all messages in the error stack.

```
Dim amap As AcadMap
Dim prj As Project
Dim i As Long
Dim strOutput As String
Dim ee As ErrorEntry
On Error GoTo ErrHandler
Set amap = ThisDrawing.Application.
GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
prj.DrawingSet.Add ("c:\\nofile.dwg")
Exit Sub
ErrHandler:
For Each ee In amap.ErrorStack
strOutput = strOutput & ee.ErrMessage & Chr(13)
Next
MsgBox strOutput
```

ErrorStack collection

A collection of ErrorEntry collections.

Each ErrorEntry collection represents AutoCAD Map Automation API error.

Contained by the AcadMap object.

Use the ErrorStack and ErrorEntry collections to manage diagnostic messages, either system- or user-generated, when an error occurs in executing the AutoCAD Map Automation API. You add your own messages to the error stack using the Add method. You get the LastErrCode property immediately after an error occurs to read the last error code generated.

The following example adds a nonexistent file to the drawing set and shows how to display all messages in the error stack.

```
Dim amap As AcadMap
Dim prj As Project
Dim i As Long
Dim strOutput As String
Dim ee As ErrorEntry
On Error GoTo ErrHandler
Set amap = ThisDrawing.Application.
GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
prj.DrawingSet.Add ("c:\\nofile.dwg")
Exit Sub
ErrHandler:
For Each ee In amap.ErrorStack
strOutput = strOutput & ee.ErrMessage & Chr(13)
Next
MsgBox strOutput
```

Expression object

Represents the expression evaluator.

Created by calling MapUtil.NewExpression.

Contained by a QueryReport collection. Can be contained by an AlterLine object.

You use expressions primarily in Report mode queries and property alteration definitions, but you can use them for general purposes, such as getting the area of all objects in a selection set.

You control the format of the query report by defining a template using expressions. Each expression defines a column of the report. When you execute the query with a report tempate, the information specified by the expressions is written to the report. For example, a report template with three expressions creates three columns, separated by commas, of information about queried objects.

For more information, see "Expressions" in AutoCAD Map UI Help.

ɔject

<u>bject Model</u> <u>See Also</u> Ssion evaluator.

lapUtil.NewExpression.

yReport collection. Can be contained by an AlterLine

primarily in Report mode queries and property alteration can use them for general purposes, such as getting the area ection set.

at of the query report by defining a template using pression defines a column of the report. When you execute ort tempate, the information specified by the expressions is For example, a report template with three expressions s, separated by commas, of information about queried

n, see "Expressions" in AutoCAD Map UI Help.

FenceBound collection

A FenceBound collection is a polygon boundary, also called a fence boundary. Created with MapUtil.NewFence to create a new FenceBound object.

collection

<u>Diject Model</u> See Also ction is a polygon boundary, also called a fence boundary.

il.NewFence to create a new FenceBound object.

MapUtil object

<u>Methods</u> Properties <u>Object Model</u> <u>See Also</u> Used to create object data tables, boundaries for queries, the expression object, and other objects.

Contained by a project.

Has methods for creating various query components such as boundary objects used by location queries. Also used to create ODFieldDefs collections that represents the schema of an object data table. ct

<u>Object Model</u> See Also t data tables, boundaries for queries, the expression object,

ect.

ating various query components such as boundary objects ries. Also used to create ODFieldDefs collections that a of an object data table.

NestedDrawings collection

<u>Methods</u> <u>Properties</u> <u>Object Model</u> <u>See Also</u> NestedDrawings represents a collection of drawings attached to an attached drawing.

The NestedDrawings collection is contained by the AttachedDrawing object, which is contained by a DrawingSet collection.

You cannot examine the objects of this class if the parent object is not active. Check the AttachedDrawing.Active property before trying to get the NestedDrawings object from the AttachedDrawing object.

You can use a NestedDrawings collection if you first attach a drawing, which has attached drawings associated with it, to a project, and then attach other drawings to that AttachedDrawing object by using a NestedDrawings collection.

ngs collection

<u>Diject Model</u> <u>See Also</u> resents a collection of drawings attached to an attached

s collection is contained by the AttachedDrawing object, y a DrawingSet collection.

the objects of this class if the parent object is not active. Drawing.Active property before trying to get the ect from the AttachedDrawing object.

dDrawings collection if you first attach a drawing, which has sociated with it, to a project, and then attach other drawings ving object by using a NestedDrawings collection.

ODFieldDef object

Methods <u>Properties</u> <u>Object Model</u> <u>See Also</u> Represents a field definition.

Contained by an ODFieldDefs collection.

Created by the ODFieldDefs.Add method.

After calling ODFieldDefs.Add to add a field definition, you define the field using the properties of this object.

object

<u>Object Model</u> <u>See Also</u> finition.

FieldDefs collection.

eldDefs.Add method.

dDefs.Add to add a field definition, you define the field of this object.

ODFieldDefs collection

<u>Methods</u> <u>Properties</u> <u>Object Model</u> <u>See Also</u> Represents the schema for an object data table.

Created by calling MapUtil.NewODFieldDefs.

Contains ODFieldDef objects.

Contained by an ODTable object.

When you add a new table to an ODTables collection, you specify its fields using this collection.

collection

<u>)bject Model</u> <u>See Also</u> na for an object data table.

lapUtil.NewODFieldDefs.

ef objects.

Table object.

⁷ table to an ODTables collection, you specify its fields

ODFieldValue object

Methods <u>Properties</u> <u>Object Model</u> <u>See Also</u> One of the fields in a record.

Contained by an ODRecord collection.

Created when the containing record is created by ODTable.CreateRecord.

Properties of this object are the value of the field and its data type. To get or set data values in a record, access the Value property of this object as follows: ODRecord.Item(n).Value.

e object

Object Model See Also I record.

Record collection.

ntaining record is created by ODTable.CreateRecord.

ect are the value of the field and its data type. To get or set rd, access the Value property of this object as follows: */*alue.

ODRecord collection

<u>Methods</u> <u>Properties</u> <u>Object Model</u> <u>See Also</u> An object data record.

Contains ODFieldValue objects.

Contained by an ODRecords collection.

Created by ODTable.CreateRecord.

For more information, click .

The following example shows how to use several properties of the Record object. First, create an object data table called SampleOD. For more information, click **2**. Next, put this code into a procedure and run it. The object ID and value of data attached to each object from the SampleOD table prints in the Immediate Window.

Dim amap As AcadMap Dim acadObj As Object Dim ODtb As ODTable Dim i As Integer Dim prj As Project Dim ODrcs As ODRecords Dim boolVal As Boolean

```
Set amap = ThisDrawing.Application. _
GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
prj.ProjectOptions.DontAddObjectsToSaveSet = True
Set ODtb = prj.ODTables.Item("SampleOD")
Set ODrcs = ODtb.GetODRecords
For Each acadObj In ThisDrawing.ModelSpace
boolVal = ODrcs.Init(acadObj, True, False)
Debug.Print ODrcs.Record.tableName
Debug.Print ODrcs.Record.ObjectID
For i = 0 To ODrcs.Record.Count - 1
Debug.Print ODrcs.Record.Item(i).Value
Next i
```

ODRecords collection

Represents object data attached to an object from tables.

Created automatically by the project.

You can attach records from more than one table, and multiple records from the same table, to a drawing object. The ODRecords collection always references a particular drawing object, which you designate by calling the ODRecords.Init method. You must explicitly release an ODRecords object when you are finished with it. See <u>ODRecords.Init()</u> for details.

ollection

<u>Diject Model</u> <u>See Also</u> ta attached to an object from tables.

y by the project.

ds from more than one table, and multiple records from the ring object. The ODRecords collection always references a bject, which you designate by calling the ODRecords.Init cplicitly release an ODRecords object when you are finished rds.Init() for details.
ODTable object

MethodsPropertiesObject ModelSee AlsoRepresents an object data table.Contained by an ODTables collection.Created by ODTables.Add.For more information, click ■.

ect

<u>bject Model</u> <u>See Also</u> data table.

Tables collection.

s.Add.

n, click 🛎.

ODTables collection

A collection of object Model See Also A collection of object data tables. Contained by a project. Created automatically by the project. For more information, click .

llection

<u>)bject Model</u> <u>See Also</u> :t data tables.

ect.

y by the project.

n, click 🛎.

Point3d object

Methods <u>Properties</u> <u>Object Model</u> <u>See Also</u> A point used in the composition of boundary objects.

Note that Point3d objects can serve as boundary objects by themselves. In that situation, whether the condition is Crossing or Inside doesn't matter. Objects returned include the following.

Lines that are coincident with the point Polylines if a straight segment is coincident Circles and arcs if the center is coincident

Note that circles and arcs are returned only if the center is coincident. Coincidence of the curve doesn't matter. More complicated figures are returned only if a vertex or control point is coincident. Ľ

Object Model See Also omposition of boundary objects.

jects can serve as boundary objects by themselves. In that e condition is Crossing or Inside doesn't matter. Objects following.

incident with the point right segment is coincident if the center is coincident

arcs are returned only if the center is coincident. urve doesn't matter. More complicated figures are returned ntrol point is coincident.

PolygonBound collection

A polygon boundary.

Created by the MapUtil.NewPolygon method.

ıd collection

<u>)bject Model</u> <u>See Also</u>

Jtil.NewPolygon method.

PolylineBound collection

A polyline boundary. <u>Methods</u> <u>Properties</u> <u>Object Model</u> <u>See Also</u>

Created by the MapUtil.NewPolyline method.

ıd collection

<u>)bject Model</u> <u>See Also</u>

Jtil.NewPolyline method.

Project object

An AutoCAD Map Project. <u>See Also</u>

Contained by the Projects collection of the application.

Contains all the elements of a project.

t

<u>)bject Model</u> <u>See Also</u>)roject.

jects collection of the application.

ients of a project.

Projects collection

Represents existing AutoCAD Map projects.

Contained by AcadMap.

Created for you when you get an AcadMap object.

This collection contains all AutoCAD Map projects. A project is an open AutoCAD Map document. When an application opens or closes an AutoCAD Map document, the Projects collection automatically adds projects to, or removes projects from, the Projects collection.

You get a project from the collection by using the AcadMap.Projects.Item method.

ction

<u>bject Model</u> <u>See Also</u> AutoCAD Map projects.

1ap.

n you get an AcadMap object.

ins all AutoCAD Map projects. A project is an open ment. When an application opens or closes an AutoCAD Projects collection automatically adds projects to, or m, the Projects collection.

m the collection by using the AcadMap.Projects.Item

Query object

<u>Methods</u> <u>Properties</u> <u>Object Model</u> <u>See Also</u> The current query.

Contained by a project.

Created automatically by the project.

Contains a QueryBranch collection called the main branch, which contains the query's conditions (QueryLeaf objects). The main branch can also contain subbranches (other QueryBranch collections), which contain grouped conditions (more QueryLeaf objects). Sub-branches can contain other sub-branches, and so on.

To assign a main branch to a query, use the Query.Define method. To get a query's main branch, use the Query.QueryBranch property.

For more information, click \blacksquare .

<u>)bject Model</u> <u>See Also</u>

ect.

y by the project.

inch collection called the main branch, which contains the JueryLeaf objects). The main branch can also contain subryBranch collections), which contain grouped conditions jects). Sub-branches can contain other sub-branches, and so

nch to a query, use the Query.Define method. To get a use the Query.QueryBranch property.

n, click 🛎.

QueryBranch collection

<u>Methods</u> <u>Properties</u> <u>Object Model</u> <u>See Also</u> Represents a grouped set of query conditions.

QueryBranch collections can contain either QueryLeaf objects, other QueryBranch objects, or both.

Solitary conditions are represented by QueryLeaf objects.

The main branch of the current query is contained by Project.CurrQuery and accessed through the Project during query creation.

For more information about queries, click \blacksquare .

collection

<u>)bject Model</u> <u>See Also</u> I set of query conditions.

ions can contain either QueryLeaf objects, other 3, or both.

re represented by QueryLeaf objects.

the current query is contained by Project.CurrQuery and Project during query creation.

n about queries, click 🗷.

QueryCategories collection

A query library. Object Model See Also

A collection of query categories, which are collections of queries.

Contained by the project.

Created by default when the project is created.

ries collection

<u>)bject Model</u> <u>See Also</u>

r categories, which are collections of queries.

)ject.

^rhen the project is created.

QueryCategory collection

A collection of queries.

Created by adding the category to the library (a QueryCategories collection).

ry collection

<u>)bject Model</u> <u>See Also</u> **eS.**

e category to the library (a QueryCategories collection).

QueryReport collection

A collection of expressions for a report template.

Contained by a query.

A report stores results of a query in a text file. You control the format of the report by defining a template using expressions. Each expression defines a column of the report, and can include dot variables that represent object properties, object data cariables that represent nongraphic information stored with the object int he drawing file, and SQL variables that represent the connection between objects and external database tables. When you execute the query with a report tempate, the information specified by the expressions is written to the report. For example, a report template with three expressions creates three columns, separated by commas, of information about queried objects.

To specify what information you want in the report, create a template. When the query with a report template executes, the output report is sent to a file.

collection

<u>bject Model</u> <u>See Also</u> ssions for a report template.

ÿ.

s of a query in a text file. You control the format of the template using expressions. Each expression defines a , and can include dot variables that represent object ta cariables that represent nongraphic information stored drawing file, and SQL variables that represent the objects and external database tables. When you execute the empate, the information specified by the expressions is For example, a report template with three expressions s, separated by commas, of information about queried

rmation you want in the report, create a template. When the emplate executes, the output report is sent to a file.
Range object

Methods <u>Properties</u> <u>Object Model</u> <u>See Also</u> A range in a range table.

A range specifies a condition to check, such as "less than 1," and a value to return if the condition is true, such as "green."

For more information, click **N**. Also go to "Using AutoCAD Map > Queries > Altering the Properties of Queried Objects > Creating a Range Table" in AutoCAD Map UI Help.

<u>Object Model</u> <u>See Also</u> ble.

ondition to check, such as "less than 1," and a value to n is true, such as "green."

n, click N. Also go to "Using AutoCAD Map > Queries > es of Queried Objects > Creating a Range Table" in Ielp.

RangeTable collection

A collection of ranges.

Contained by the RangeTables collection of a project.

Created by RangeTables.Add followed by calls to RangeTable.Add as needed to build the collection.

Range tables allow conditional property alterations for example, an alteration that colors an object green if its width is less than 1, or colors it blue otherwise. To specify a conditional alteration, first define a range table by building a RangeTable collection, and then use a range table expression instead of an explicit value or expression when you set the Value parameter of a property alteration (or the TextValue parameter of a text alteration, or the Pattern property of a Hatch alteration).

For more information, click . Also, go to "Using AutoCAD Map > Queries > Altering the Properties of Queried Objects > Creating a Range Table" in AutoCAD Map UI Help.

This example is based on the sample code for altering queried objects. To go to the sample code, click . The following example shows how to set up a range table called safewater. The query using the range table moves water bodies less than 10 feet from the WATER layer to the STREAM layer.

Dim rngtb As RangeTable Dim rng As Range Dim altlLay As AlterLine

RangeTables collection

<u>Methods</u> <u>Properties</u> <u>Object Model</u> <u>See Also</u> Range library, a collection of range tables defined in the project.

Contained by the project.

Created automatically by the project.

For more information, click . Also, go to "Using AutoCAD Map > Queries > Altering the Properties of Queried Objects > Creating a Range Table" in AutoCAD Map UI Help.

collection

<u>Diject Model</u> See Also ection of range tables defined in the project.

)ject.

```
y by the project.
```

```
n, click N. Also, go to "Using AutoCAD Map > Queries > es of Queried Objects > Creating a Range Table" in Ielp.
```

SavedQuery object

Methods <u>Properties</u> <u>Object Model</u> <u>See Also</u> A query saved in a query library.

Created by the QueryCategory.Add, QueryCategory.AddFromFile, or Query.Save method.

You can save internal queries or queries from an external file in the query library. The properties of the saved query name it and indicate whether it resides in an external file. If the query is external, another property represents its path.

The following example shows how to create, retrieve, and execute a saved query.

```
Dim atdr As AttachedDrawing
Dim amap As AcadMap
Dim prj As Project
Dim qry As Query
Dim mainqrybr As QueryBranch
Dim qrylf As QueryLeaf
Dim boolVal As Boolean
Dim i As Integer, cqryct As Integer
Dim j As Integer, cqry As Integer
Dim sqry As SavedQuery
```

```
Set amap = ThisDrawing.Application. _
GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
prj.CurrQuery.Clear
Set atdr = prj.DrawingSet.Add("MAPTUT:\\citymap7.dwg")
Set qry = prj.CurrQuery
qry.Clear
Set mainqrybr = qry.QueryBranch
Set qrylf = mainqrybr.Add(kLocationCondition, kOperatorAnd)
qrylf.SetLocationCond kLocationInside, prj.MapUtil.NewLocationAll
qry.Mode = kQueryDraw
qry.Define mainqrybr
qry.Save False, 0, "DrawingCategory", "MyInternalquery", __
"Qry in library", ""
For i = 0 To prj.QueryCategories.Count - 1
If prj.QueryCategories(i).Name = "DrawingCategory" Then
cqryct = i
```

```
End If
Next i
For j = 0 To prj.QueryCategories.Item(cqryct).Count - 1
If prj.QueryCategories.Item(cqryct).Item(j).Name = "MyInternalquery" Then
cqry = j
End If
Next j
Set sqry = prj.QueryCategories.Item(cqryct).Item(cqry)
prj.CurrQuery.Load (sqry)
prj.CurrQuery.Execute
ThisDrawing.Application.ZoomExtents
```

SaveSet collection

A set of changed, deleted, or new objects to be saved back to the source drawings.

Contained by a project.

SaveSet is a collection of new objects in a drawing that are saved in the default file of a project if none other is specified. A SaveSet collection also manages objects that you save into one or more attached drawings that are active.

ction

<u>bject Model</u> <u>See Also</u> leted, or new objects to be saved back to the source

ect.

on of new objects in a drawing that are saved in the default ne other is specified. A SaveSet collection also manages into one or more attached drawings that are active.

SystemOptions object

Methods <u>Properties</u> <u>Object Model</u> <u>See Also</u> Represents options for the AutoCAD Map application.

Contained by the AcadMap object.

Created automatically when you call acadApp.GetInterfaceObject to create AcadMap.

Options for individual AutoCAD Map projects are represented by ProjectOptions objects. For more information about options, go to Using AutoCAD Map > Setting Options > Setting System Options in AutoCAD Map UI Help.

SystemOptions Properties

The SystemOptions object has no methods. Its properties are listed in six groups below:

Workspace Options System Options Database Options MultiUser Options DatabaseDrivers Options Additional Options

Workspace Options

CheckDrawings property	Boolean	Read-Write
DrawingsVisible property	Boolean	Read Only
CheckQueryLibrary property	Boolean	Read-Write
QueryLibraryVisible property	Boolean	Read Only
CheckDatabases property	Boolean	Read-Write
DatabasesVisible property	Boolean	Read Only
CheckTables property	Boolean	Read-Write
TablesVisible property	Boolean	Read Only

CheckQueries property	Boolean	Read-Write
QueriesVisible property	Boolean	Read Only
CheckTopologies property	Boolean	Read-Write
TopologiesVisible property	Boolean	Read Only
CheckLTs property	Boolean	Read-Write
LTsVisible property	Boolean	Read Only
ShowWSpaceOnStartup property	Boolean	Read-Write
WspaceDockingView property	Boolean	Read-Write
ShowOPMOnStartup property	Boolean	Read-Write
CheckClasses property	Boolean	Read-Write
ClassesVisible property	Boolean	Read Only

System Options

NumberofOpenDwgs property	Long
LogFileActive property	Boolean
LogFileName property	String
LogMessageLevel property	Long
QueryFileDirectory property	String

Database Options

DisplayTabsInSingleView property	Boolean
OpenDataViewReadOnly property	Boolean
SaveDataViewFmtChanges property	Boolean
KeepDataViewOnTop property	Boolean

MultiUser Options

ForceUserLogin property	Boolean	This property cannot be set unless the current user is logged in as a superuser, and no drawings are attached.
EnableObjectLocking property	Boolean	

DatabaseDrivers Options

DriverForDbaseDbf property	String
DriverForExcelXls property	String
DriverForParadoxDb property	String

Additional Options

ActivePreferencePage property	Long
IncludeCoordsysInDrawing property	Boolean
MapDistUnits property	String

WindowBound object

Methods <u>Properties</u> <u>Object Model</u> <u>See Also</u> A rectangle boundary.

Created by the MapUtil.NewWindow method.

For information about modifying queries, click 🔳

The following example uses WindowBound to create the main branch of a query of all objects within the extents of the attached drawing.

```
Dim amap As AcadMap
Dim prj As Project
Dim qry As Query
Dim atdr As AttachedDrawing
Dim mainqrybr As QueryBranch
Dim qrylf As QueryLeaf
Dim mapu As MapUtil
Dim wind As WindowBound
Dim dblary As Variant
```

```
Set amap = ThisDrawing.Application. _

GetInterfaceObject("AutoCADMap.Application")

Set prj = amap.Projects(ThisDrawing)

Set qry = prj.CurrQuery

qry.Clear

Set atdr = prj.DrawingSet.Add("MAPTUT:\\citymap7.dwg")

Set mainqrybr = qry.QueryBranch

Set qrylf = mainqrybr.Add(kLocationCondition, kOperatorAnd)

dblary = prj.DrawingSet.Item("MAPTUT:\\citymap7.dwg").Extents

Set mapu = prj.MapUtil

Set wind = mapu.NewWindow( _

mapu.NewPoint3d(dblary(0), dblary(1), 0), _

mapu.NewPoint3d(dblary(2), dblary(3), 0))

qrylf.SetLocationCond kLocationInside, wind
```

Aliases.Add method

Aliases collection See Also Defines a drive alias.

Add(AliasName As String, AliasPath As String) As Alias

Returns the new drive alias.

AliasName

Name of the alias.

The name must be unique, use only alphanumeric characters (including hyphen and underscore), contain no spaces or colons, start with a character, and not exceed 31 characters.

AliasPath

An existing path that the alias represents.

If the path represented by AliasPath does not exist, the Add method does not create an alias. Trying to add an alias of the same name as an existing one causes an error.

The following example checks for an alias of the same name before adding an alias for the Temp directory called SHAREDDRIVE.

Dim als As Alias Dim amap As AcadMap Dim cAls As Integer, i As Integer Dim strAlsName As String Dim boolAlsOK As Boolean

```
boolAlsOK = True
strAlsName = "SHAREDDRIVE"
Set amap = ThisDrawing.Application. __
GetInterfaceObject("AutoCADMap.Application")
cAls = amap.aliases.Count
For i = 0 To cAls - 1
Set als = amap.aliases.Item(i)
If als.Name = strAlsName Then
Debug.Print "Alias already exists"
boolAlsOK = False
End If
Next i
```

If boolAlsOK = True Then Set als = amap.aliases.Add(strAlsName, "C:\\Temp") End If

Aliases.FindByPath method

Gets a drive alias.

FindByPath(RealPath As String) As Alias

Returns the alias.

RealPath

The actual path represented by an existing alias.

Use a second backslash to delimit each backslash in the path.

The following example assumes the existence of a drive alias for C:\temp. The example shows how to find and print the name of the alias in the Immediate Window.

Dim als As Alias Dim amap As AcadMap

Set amap = ThisDrawing.Application. _ GetInterfaceObject("AutoCADMap.Application") Set als = amap.Aliases.FindByPath("C:\\Temp") Debug.Print als.Name

3yPath method

<u>Also</u>

Path As String) As Alias

epresented by an existing alias.

ash to delimit each backslash in the path.

ple assumes the existence of a drive alias for C:\temp. The to find and print the name of the alias in the Immediate

∕Лар

```
ving.Application. _
'AutoCADMap.Application")
es.FindByPath("C:\\Temp")
ie
```

Aliases.Item method

Aliases collection See Also Gets an alias.

Item(Index As Variant) As Alias

Returns the alias.

Index

Index, starting at 0, or name of an alias in the collection.

The following example shows how to display each item in the aliases collection.

Dim als As Alias Dim cAls As Integer, i As Integer Dim amap As AcadMap Dim strOutput As String

```
Set amap = ThisDrawing.Application. __
GetInterfaceObject("AutoCADMap.Application")
cAls = amap.Aliases.Count
For i = 0 To cAls - 1
Set als = amap.Aliases.Item(i)
strOutput = strOutput & als.Name & "=" & als.Directory & Chr(13)
Next i
MsgBox strOutput
```

method

<u>Also</u>

riant) As Alias

0, or name of an alias in the collection.

ple shows how to display each item in the aliases collection.

', i As Integer ⁄Iap ring

*w*ing.Application. _ 'AutoCADMap.Application'') .Count

es.Item(i) It & als.Name & "=" & als.Directory & Chr(13)

Aliases.Remove method

Aliases collection See Also Removes a drive alias.

Remove(Index As Variant) As Boolean

Returns True on success.

Index

Index, starting at 0, or name of an alias in the collection.

Each project comes with a default alias called C:=C. Removing this alias is not recommended and causes an error when you run the following example. This example creates an alias called SHAREDDRIVE, lists all defined alises, and then removes SHAREDDRIVE.

```
Dim als As Alias
Dim amap As AcadMap
Dim cAls As Integer, i As Integer
Dim strOutput As String
Dim strAlsName As String
Dim boolAlsOK As Boolean
```

```
boolAlsOK = True
strAlsName = "SHARED2DRIVE"
Set amap = ThisDrawing.Application.
GetInterfaceObject("AutoCADMap.Application")
cAls = amap.aliases.Count
'Code to check uniqueness of alias name goes here. See Aliases.Add
Set als = amap.aliases.Add(strAlsName, "C:\\Temp")
cAls = amap.aliases.Count
For i = 0 To cAls - 1
Set als = amap.aliases.Item(i)
strOutput = strOutput & als.Name & "=" & als.Directory & Chr(13)
Next i
MsgBox strOutput
cAls = amap.aliases.Count
For i = 0 To cAls - 1
Set als = amap.aliases.Item(i)
If als.Name = strAlsName Then
amap.aliases.Remove (als.Name)
```

Debug.Print "Removed " & strAlsName End If Next i

AlterLines.Add method

AlterLines collection Examples See Also Adds a property alteration to the property alteration definition.

Add(AlterType As EAlterationType, _ Expression As String _) As LPDISPATCH

Returns an AlterLine, HatchDef, or TextDef, depending on AlterType.

AlterType

Specifies the type of property alteration.

Expression

An expression, evaluated at the time of query execution, to alter properties.

The result of the expression must match the data type of the queried property. If you pass AlterationElevation as the AlterType, the Expression must evaluate to a point. If you pass kAlterationHeight, kAlterationRotation, kAlterationWidth, or kAlterationThickness, Expression must evaluate to a real. For all other values of AlterType, Expression must evaluate to a string. For more information about expressions, see "Expressions" in AutoCAD Map UI Help.

You can call this method to add a hatch or text alteration to the collection by passing the KAlterationHatch or kAlterationTextEntity constants in the first argument, as shown in the example. The API then uses the default hatch or text entity definitions. To use other definitions, call the AddHatchDef and AddTextDef methods instead of this method.

The following excerpt from a larger example defines a query that alters the color of drawing objects and adds a text entity, the name, to water bodies on the map. For the complete example, click \blacksquare .

Dim qry As Query Dim altls As AlterLines Dim altl As AlterLine Dim txtdf As TextDef ' Create a query -- code not shown Set altls = qry.AlterProp altls.RemoveAll Set altl = altls.Add(kAlterationColor, "RED") Set txtdf = altls.Add(__ kAlterationTextEntity, __ ":NAME@WATER_BODIES") qry.AlterTag = True ' Define and execute the query to alter properties -- code not shown
AlterLines.AddHatchDef method

AlterLines collection See Also Adds a hatch alteration to the property alteration definition.

AddHatchDef(_ Pattern As String, _ Scale As String, _ Rotation As String, _ Layer As String, _ Color As String _) As HatchDef

Returns a hatch definition.

Pattern

Name or expression of the hatch pattern

Scale

Hatch scale or expression

Rotation

Hatch rotation or expression

Layer

Layer or expression where the hatch is drawn.

Color

Hatch color or representative expression

The following example builds on sample code for altering queried objects. For more information, click . Add the following code to the example just before qry.AlterTag = True. This code adds a solid, cyan hatch pattern to objects on the WATER layer.

Dim hatdf As HatchDef Set hatdf = altls.AddHatchDef("Solid", "600", "0", "WATER", "CYAN")

AlterLines.AddTextDef method

AlterLines collection See Also Adds a text alteration to the property alteration definition.

AddTextDef(_ TextValue As String, _ Height As String, _ InsertPoint As String, _ Justification As String, _ TextStyle As String, _ Layer As String, _ Color As String, _ Rotation As String _) As TextDef

Returns a text definition.

TextValue

Text value or expression

Height

Text height or expression

InsertPoint

The location for insertion of the text or expression

Justification

Text alignment or expression

Text Style

Text style or expression

Layer

Layer where the text is drawn or expression

Color

Text color or expression

Rotation

Text rotation or expression

This example builds on sample code for altering queried objects. For more information, click **B**. The following code adds a text entity in Arial font that is larger than the text created by the original sample code. Comment out the call to Alterlines.Add in the sample and add the call to Alterlines.AddTextDef as shown here:

```
'Set txtdf = altls.Add( _
' kAlterationTextEntity, _
' ":NAME@WATER_BODIES")
Set txtdf = altls.AddTextDef(":NAME@WATER_BODIES", "200", _
".LABELPT", "CENTER", "Arial", "Water", "Bylayer", "0")
```

AlterLines.Item method

<u>AlterLines collection</u> Gets an alteration in the property alteration definition.

Item(Index As Long) As LPDISPATCH

Returns an alteration.

Index

The index of an alteration, starting at 0.

The following code builds on the sample code for altering queried objects. Using Alterlines.Item, this code gets the property alteration for color and prints its value. Insert the code just before the call to qry.Define in the sample code for altering queried object. To go to the sample code, click click **B**.

```
Dim i As Integer
For i = 0 To prj.CurrQuery.AlterProp.Count - 1
Set altl = altls.Item(i)
If altl.Property = kAlterationColor Then
Debug.Print altl.Value
End If
Next
```

em method

l.

the property alteration definition.

```
ng) As LPDISPATCH
```

lteration, starting at 0.

ouilds on the sample code for altering queried objects. Using code gets the property alteration for color and prints its e just before the call to qry.Define in the sample code for ct. To go to the sample code, click click **N**.

```
Query.AlterProp.Count - 1
)
lterationColor Then
Ie
```

AlterLines.Remove method

<u>AlterLines collection</u> Removes a property alteration from the property alteration definition.

Remove(Index As Long) As Boolean

Returns True on success.

Index

Index of an alteration, starting at 0.

The following code removes the property alteration that colors queried objects red in the sample code for altering queried objects. Insert the code just before the call to qry.Define in the sample code. To go to the sample code, click **D**.

```
Dim i As Integer
For i = 0 To prj.CurrQuery.AlterProp.Count
Set altl = altls.Item(i)
If altl.Property = kAlterationColor Then
prj.CurrQuery.AlterProp.Remove (i)
Exit For
End If
Next i
```

emove method

alteration from the property alteration definition.

; Long) As Boolean

cess.

ation, starting at 0.

removes the property alteration that colors queried objects le for altering queried objects. Insert the code just before the the sample code. To go to the sample code, click **N**.

Query.AlterProp.Count) lterationColor Then 'rop.Remove (i)

AlterLines.RemoveAll method

<u>AlterLines collection</u> <u>Example</u> Removes all alterations from the property alteration definition.

RemoveAll() As Boolean

Returns True on success.

The following excerpt from a larger example first clears the current query and then clears its property alterations by calling AlterLines.RemoveAll. For the complete example, click **B**.

Dim amap As AcadMap

Dim qry As Query

Dim altls As AlterLines

```
Set amap = ThisDrawing.Application. _
GetInterfaceObject("AutoCADMap.Application")
Set qry = amap.Projects.Item(ThisDrawing).CurrQuery
qry.Clear
' Define query main branch and conditions -- not shown
Set altls = qry.AlterProp
altls.RemoveAll
```

emoveAll method

<u>xample</u> ons from the property alteration definition.

Boolean

cess.

pt from a larger example first clears the current query and ty alterations by calling AlterLines.RemoveAll. For the lick ■.

∕Іар

nes

wing.Application. _
'AutoCADMap.Application")
>cts.Item(ThisDrawing).CurrQuery

branch and conditions -- not shown 'rop

AttachedDrawing.AddAllToSSet method

AttachedDrawing object See Also Adds to a save set all the objects queried from a drawing.

AddAllToSSet() As Boolean

Returns True on success.

The following example saves all queried objects to a save set.

```
Dim amobj as AcadMap
Dim pobj As AutoCADMap.Project
Dim dsobj as DrawingSet
Dim adobj as AttachedDrawing
Dim retval as Boolean
Set amobj = acadApp.GetInterfaceObject("AutoCADMap.Application")
Set pobj = amobj.Projects(ThisDrawing)
Set dsobj = pobj.DrawingSet
Set adobj = dsobj.Item(5)
retval = adobj.AddAllToSSet
If retval = True Then
MsgBox "Queried drawing objects saved to a set."
Else
MsgBox "Queried drawing objects not saved to a set."
```

wing.AddAllToSSet method

l the objects queried from a drawing. As Boolean cess. ple saves all queried objects to a save set. Мар ADMap.Project ıgSet edDrawing in p.GetInterfaceObject("AutoCADMap.Application") >jects(ThisDrawing) wingSet m(5) llToSSet L rawing objects saved to a set."

rawing objects not saved to a set."

AttachedDrawing.CountInSSet method

AttachedDrawing object See Also Counts the objects in a save set.

CountInSSet(Qualifier As ESaveSetObjectType) As Long

Returns the count.

Qualifier

The types of objects you want to count, one or more ESaveSetObjectType constants.

You can combine constants to specify more than one type. For example, to get the number of queried objects that have not been deleted plus the number of new objects, the Qualifier argument is kQueriedExisted + kNewlyCreated.

The following example gets the number of queried objects that have been deleted.

Dim amobj as AcadMap Dim pobj As AutoCADMap.Project Dim dsobj as DrawingSet Dim adobj as AttachedDrawing Dim objnum as Long Set amobj = acadApp.GetInterfaceObject("AutoCADMap.Application") Set pobj = amobj.Projects(ThisDrawing) Set dsobj = pobj.DrawingSet Set adobj = dsobj.Item(5) objnum = adobj.CountInSSet(kQueriedErased) MsgBox "The number of locked objects is: " & CStr(objnum)

wing.CountInSSet method

<u>See Also</u> I a save set.

lifier As ESaveSetObjectType) As Long

ects you want to count, one or more ESaveSetObjectType

nstants to specify more than one type. For example, to get ed objects that have not been deleted plus the number of new c argument is kQueriedExisted + kNewlyCreated.

ple gets the number of queried objects that have been

Map ADMap.Project gSet edDrawing p.GetInterfaceObject("AutoCADMap.Application") ojects(ThisDrawing) wingSet m(5) ntInSSet(kQueriedErased) er of locked objects is: " & CStr(objnum)

AttachedDrawing.GetTableList method

<u>AttachedDrawing object</u> <u>See Also</u> Gets symbol table information to use in queries.

GetTableList(PropertyIndex As ETableType) As Variant

Returns symbol table information for the specified property, an array of String.

PropertyIndex

A symbol table.

AutoCAD Map stores non-graphical information such as block definitions, layers, groups, and text styles in symbol tables. Any source drawing or project can have symbol tables.

The following example gets information from a symbol table.

Dim amobj as AcadMap Dim pobj As AutoCADMap.Project Dim dsobj as DrawingSet Dim adobj as AttachedDrawing Dim blockset(1 to 100) as String Set amobj = acadApp.GetInterfaceObject("AutoCADMap.Application") Set pobj = amobj.Projects(ThisDrawing) Set dsobj = pobj.DrawingSet Set adobj = dsobj.Item(5) blockset = adobj.GetTableList(kBlockTable) wing.GetTableList method

formation to use in queries.

pertyIndex As ETableType) As Variant

e information for the specified property, an array of String.

es non-graphical information such as block definitions, ext styles in symbol tables. Any source drawing or project les.

ple gets information from a symbol table.

Map ADMap.Project ngSet edDrawing 00) as String p.GetInterfaceObject("AutoCADMap.Application") ojects(ThisDrawing) wingSet m(5) TableList(kBlockTable)

AttachedDrawing.QuickView method

<u>AttachedDrawing object</u> Displays a quick view of an attached drawing.

QuickView() As Boolean

Returns True if a drawing can be shown in quick view.

The following example sets the drawing for quick view.

```
Dim amobj as AcadMap
Dim pobj As AutoCADMap.Project
Dim dsobj as DrawingSet
Dim adobj as AttachedDrawing
Dim retval as Boolean
Set amobj = acadApp.GetInterfaceObject("AutoCADMap.Application")
Set pobj = amobj.Projects(ThisDrawing)
Set dsobj = pobj.DrawingSet
Set adobj = dsobj.Item(5)
retval = adobj.QuickView
If retval = True Then
MsgBox "The drawing can be displayed in quick view."
Else
MsgBox "The drawing cannot be displayed in quick view."
```

wing.QuickView method

w of an attached drawing.

Boolean

drawing can be shown in quick view. ple sets the drawing for quick view. Vap ADMap.Project gSet edDrawing n p.GetInterfaceObject("AutoCADMap.Application") ojects(ThisDrawing) wingSet m(5) View 1 ing can be displayed in quick view."

ing cannot be displayed in quick view."

AttachedDrawing.RemoveAllFromSSet method

AttachedDrawing object See Also Removes from a SaveSet all objects queried from a drawing.

RemoveAllFromSSet() As Boolean

Returns True on success.

The following example removes queried objects from a save set.

```
Dim amobj as AcadMap
Dim pobj As AutoCADMap.Project
Dim dsobj as DrawingSet
Dim adobj as AttachedDrawing
Dim retval as Boolean
Set amobj = acadApp.GetInterfaceObject("AutoCADMap.Application")
Set pobj = amobj.Projects(ThisDrawing)
Set dsobj = pobj.DrawingSet
Set adobj = dsobj.Item(5)
retval = adobj.RemoveAllFromSSet
If retval = True Then
MsgBox "Queried objects removed from save set."
Else
MsgBox "Queried objects not removed from save set."
```

wing.RemoveAllFromSSet method

```
reSet all objects queried from a drawing.
Set() As Boolean
cess.
ple removes queried objects from a save set.
Мар
ADMap.Project
ıgSet
edDrawing
ın
p.GetInterfaceObject("AutoCADMap.Application")
>jects(ThisDrawing)
wingSet
m(5)
veAllFromSSet
L
bjects removed from save set."
```

bjects not removed from save set."

AttachedDrawing.Save method

AttachedDrawing object See Also Saves the attached drawing.

Save() As Boolean

Returns True on success.

Note Unlike the Save method, which saves to disk, the save-back methods of attached drawings (for example, the AttachedDrawing.SaveNewObjs method) save only to memory. Use AttachedDrawing.Save after calling a save-back method to make save backs persistent.

The following example shows how to save an attached drawing.

Dim amobj as AcadMap Dim pobj As AutoCADMap.Project Dim dsobj as DrawingSet Dim adobj as AttachedDrawing Dim retval as Boolean Set amobj = acadApp.GetInterfaceObject("AutoCADMap.Application") Set pobj = amobj.Projects(ThisDrawing) Set dsobj = pobj.DrawingSet Set adobj = dsobj.Item(5) retval = adobj.Save If retval = True Then MsgBox "The drawing has been saved." Else MsgBox "The drawing has not been saved."

wing.Save method

See Also rawing.

ın

cess.

e method, which saves to disk, the save-back methods of or example, the AttachedDrawing.SaveNewObjs method)
r. Use AttachedDrawing.Save after calling a save-back
e backs persistent.

ple shows how to save an attached drawing.

Map ADMap.Project IgSet edDrawing In p.GetInterfaceObject("AutoCADMap.Application") ojects(ThisDrawing) wingSet m(5)

```
ing has been saved."
```

ing has not been saved."

AttachedDrawing.SaveNewObjs method

Saves newly created objects to memory.

SaveNewObjs(IdArray As Variant) As Boolean

Returns True on success.

IdArray

IDs of the objects to save as an array of Longs.

Note Because this method saves only to memory and not to disk, use the AttachedDrawing.Save method afterward to make the changes persistent.

The following example saves newly created objects.

```
Dim amobj as AcadMap
Dim pobj As AutoCADMap.Project
Dim dsobj as DrawingSet
Dim adobj as AttachedDrawing
Dim retval as Boolean
Set amobj = acadApp.GetInterfaceObject("AutoCADMap.Application")
Set pobj = amobj.Projects(ThisDrawing)
Set dsobj = pobj.DrawingSet
Set adobj = dsobj.Item(5)
retval = adobj.SaveNewObjs(12345)
If retval = True Then
MsgBox "New object saved to memory."
Else
MsgBox "New object not saved to memory."
```

ving.SaveNewObjs method

```
<u>See Also</u>
objects to memory.
```

```
Array As Variant) As Boolean
```

cess.

s to save as an array of Longs.

ethod saves only to memory and not to disk, use the ve method afterward to make the changes persistent.

ple saves newly created objects.

Map ADMap.Project 1gSet edDrawing 1n p.GetInterfaceObject("AutoCADMap.Application") ojects(ThisDrawing) wingSet m(5) lewObjs(12345) 1 ct saved to memory."

ct not saved to memory."

AttachedDrawing.SaveObjectsByArea method

AttachedDrawing object See Also Saves objects that are within or crossing the save-back extents of the attached drawing.

SaveObjectsByArea() As Boolean

Returns True on success.

Note Because this method saves only to memory and not to disk, use the AttachedDrawing.Save method afterward to make the changes persistent.

The following example saves objects within extent boundaries.

```
Dim amobj as AcadMap
Dim pobj As AutoCADMap.Project
Dim dsobj as DrawingSet
Dim adobj as AttachedDrawing
Dim retval as Boolean
Set amobj = acadApp.GetInterfaceObject("AutoCADMap.Application")
Set pobj = amobj.Projects(ThisDrawing)
Set dsobj = pobj.DrawingSet
Set adobj = dsobj.Item(5)
retval = adobj.SaveObjectsByArea
If retval = True Then
MsgBox "Saved the objects within extent boundaries."
Else
MsgBox "Did not save the objects within extent boundaries."
End If
```
wing.SaveObjectsByArea method

e within or crossing the save-back extents of the attached

'ea() As Boolean

cess.

ethod saves only to memory and not to disk, use the ve method afterward to make the changes persistent.

ple saves objects within extent boundaries.

```
Map
ADMap.Project
ngSet
edDrawing
n
p.GetInterfaceObject("AutoCADMap.Application")
ojects(ThisDrawing)
wingSet
m(5)
bjectsByArea
1
objects within extent boundaries."
```

we the objects within extent boundaries."

AttachedDrawing.SaveObjectsByLayer method

AttachedDrawing object See Also Saves objects to drawing layers that match layers in an active drawing.

SaveObjectsByLayer() As Boolean

Returns True on success.

Note Because this method saves only to memory and not to disk, use the AttachedDrawing.Save method afterward to make the changes persistent.

```
Dim amobj as AcadMap
Dim pobj As AutoCADMap.Project
Dim dsobj as DrawingSet
Dim adobj as AttachedDrawing
Dim retval as Boolean
Set amobj = acadApp.GetInterfaceObject("AutoCADMap.Application")
Set pobj = amobj.Projects(ThisDrawing)
Set dsobj = pobj.DrawingSet
Set adobj = dsobj.Item(5)
retval = adobj.SaveObjectsByLayer
If retval = True Then
MsgBox "Saved the objects that matched the layers."
Else
MsgBox "Did not save the objects that matched the layers."
```

wing.SaveObjectsByLayer method

ving layers that match layers in an active drawing.

yer() As Boolean

cess.

ethod saves only to memory and not to disk, use the ve method afterward to make the changes persistent.

```
Map
ADMap.Project
ngSet
edDrawing
n
p.GetInterfaceObject("AutoCADMap.Application")
ojects(ThisDrawing)
wingSet
m(5)
bjectsByLayer
1
objects that matched the layers."
```

we the objects that matched the layers."

AttachedDrawing.SaveQueriedObjects method

<u>AttachedDrawing object</u> See Also Gets the objects queried from an attached drawing and saves them to memory.

SaveQueriedObjects() As Boolean

Saves images of objects from a query of the attached drawing.

Note Because this method saves only to memory and not to disk, use the AttachedDrawing.Save method afterward to make the changes persistent.

The following example shows how to save queried objects.

```
Dim amobj as AcadMap
Dim pobj As AutoCADMap.Project
Dim dsobj as DrawingSet
Dim adobj as AttachedDrawing
Dim retval as Boolean
Set amobj = acadApp.GetInterfaceObject("AutoCADMap.Application")
Set pobj = amobj.Projects(ThisDrawing)
Set dsobj = pobj.DrawingSet
Set adobj = dsobj.Item(5)
retval = adobj.SaveQueriedObjects
If retval = True Then
MsgBox "Objects saved to memory."
Else
MsgBox "Objects not saved to memory."
```

wing.SaveQueriedObjects method

ried from an attached drawing and saves them to memory.

cts() As Boolean

ects from a query of the attached drawing.

ethod saves only to memory and not to disk, use the ve method afterward to make the changes persistent.

ple shows how to save queried objects.

```
Map
ADMap.Project
ngSet
edDrawing
in
p.GetInterfaceObject("AutoCADMap.Application")
ojects(ThisDrawing)
wingSet
m(5)
ueriedObjects
i
ved to memory."
```

ot saved to memory."

BufferFenceBound.Add method

<u>BufferFenceBound collection</u> Adds a point to a BufferFenceBound collection.

Add(X As Double, _ Y As Double, _ Z As Double _) As Point3d

Returns a Point3d object.

X, Y, Z

Coordinates of the point to add

The Z coordinate is ignored.

3ound.Add method

<u>ction</u> fferFenceBound collection.

e, _

)ject.

e point to add gnored.

BufferFenceBound.Item method

<u>BufferFenceBound collection</u> Gets a point from a BufferFenceBound collection.

Item(PointID As Long) As Point3d

Returns a Point3d object.

PointID

The index of the point to get, starting from 0.

Sound.Item method

<u>ction</u> 3ufferFenceBound collection.

Long) As Point3d

)ject.

point to get, starting from 0.

BufferFenceBound.Remove method

BufferFenceBound collection Removes a point from a BufferFenceBound collection.

Remove(PointID As Long) As Boolean

Returns True on success.

PointID

The index of the point to remove, starting at zero.

Sound.Remove method

<u>ction</u> m a BufferFenceBound collection.

As Long) As Boolean

cess.

point to remove, starting at zero.

BufferPolylineBound.Item method

<u>BufferPolylineBound collection</u> Gets a point from a BufferPolylineBound collection.

Item(PointID As Long) As Object

Returns the point.

PointID

The index of the point to get, starting at 0.

eBound.Item method

<u>lection</u> 3ufferPolylineBound collection.

Long) As Object

point to get, starting at 0.

ClosedPolylineBound.Item method

<u>ClosedPolylineBound collection</u> Gets a point from a ClosedPolylineBound collection.

Item(PointID As Long) As Point3d

Returns the point.

PointID

The index of the point to get, starting at 0.

neBound.Item method

<u>llection</u> ClosedPolylineBound collection.

Long) As Point3d

point to get, starting at 0.

DrawingSet.Add method

Attaches a drawing to the project.

Add(DwgName As String) As AttachedDrawing

Returns the attached drawing.

DwgName

The alias path and file name of the drawing to be attached. For example, "ALIASNAME:\\DirName\\DwgName.dwg".

The following example attaches a drawing.

Dim prj As Project Dim drset As DrawingSet Dim atdr As AttachedDrawing Dim amap As AcadMap

Set amap = ThisDrawing.Application. _
GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
Set drset = prj.DrawingSet
Set atdr = drset.Add("ALIASNAME:\\DirName\\DwgName.dwg")

Add method

Example See Also o the project.

s String) As AttachedDrawing

drawing.

d file name of the drawing to be attached. For example, \DirName\\DwgName.dwg''.

ple attaches a drawing.

ngSet dDrawing ⁄Iap

wing.Application. _
'AutoCADMap.Application")
cts(ThisDrawing)
ingSet
("ALIASNAME:\\DirName\\DwgName.dwg")

DrawingSet.GetTableList method

<u>BrawingSet collection</u> <u>See Also</u> Gets symbol table information to use in queries.

GetTableList(PropertyIndex As ETableType) As Variant

Returns the specified symbol table information, an array of String.

PropertyIndex

A symbol table.

AutoCAD Map stores non-graphical information such as block definitions, layers, groups, and text styles in symbol tables. Any source drawing or project can have symbol tables. Use UBound to find the upper limit of the array returned by this function. Check that the array is not empty before you use it, as shown in the example, which displays the list of layers in the symbol table of a drawing.

```
Dim amap As AcadMap
Dim prj As Project
Dim strArray As Variant
Dim i As Integer
Dim strOutput As String
Set amap = ThisDrawing.Application. _
GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
strArray = prj.DrawingSet.GetTableList(kLayerTable)
If UBound(strArray) > -1 Then
  For i = 0 To UBound(strArray)
     strOutput = strOutput & strArray(i) & Chr(13)
  Next i
  MsgBox strOutput
Else
  MsgBox "No tables found."
End If
```

GetTableList method

See Also formation to use in queries. DertyIndex As ETableType) As Variant

l symbol table information, an array of String.

es non-graphical information such as block definitions, ext styles in symbol tables. Any source drawing or project les. Use UBound to find the upper limit of the array returned eck that the array is not empty before you use it, as shown in lisplays the list of layers in the symbol table of a drawing.

```
Application. _
oCADMap.Application")
(ThisDrawing)
et.GetTableList(kLayerTable)
1 Then
strArray)
tput & strArray(i) & Chr(13)
```

und."

DrawingSet.Item method

<u>DrawingSet collection</u> <u>Example</u> <u>See Also</u> Gets an attached drawing from a drawing set.

Item(AttachedDwg As Variant) As AttachedDrawing

Returns the attached drawing.

AttachedDwg

Either the index, starting at 0, of the drawing in the DrawingSet collection, or the full path and file name of the drawing in this form: alias or drive + path + filename.

The following example attaches two drawings to the drawing set, gets one of the items in the set by passing the alias, and gets the other item by passing the index of the drawing in the set.

Dim amap As AcadMap Dim prj As Project Dim drset As DrawingSet Dim atdr As AttachedDrawing Dim cDrset As Integer, i As Integer

```
Set amap = ThisDrawing.Application. _
GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
Set drset = prj.DrawingSet
Set atdr = drset.Add("ALIASNAME:\\DirName\\DwgName1.dwg")
Set atdr = drset.Add("ALIASNAME:\\DirName\\DwgName2.dwg")
Set atdr = drset.Item("ALIASNAME:\\DirName\\DwgName2.dwg")
Debug.Print "Found "; atdr.AliasPath
```

```
cDrset = drset.Count
For i = 0 To cDrset - 1
Set atdr = drset.Item(i)
If atdr.AliasPath = "ALIASNAME:\\DirName\\DwgName2.dwg" Then
Debug.Print "Found " & "DwgName2.dwg"
End If
Next i
```

DrawingSet.Remove method

DrawingSet collection See Also Detaches an attached drawing.

Remove(AttachedDwg As Variant) As Boolean

Returns True on success.

AttachedDwg

Either the index, starting at 0, of the drawing in the DrawingSet collection, or the full path and file name of the drawing to be detached.

The following example attaches two drawings, detaches the first one using the path, and detaches the other one using the item number.

```
Dim prj As Project
Dim drset As DrawingSet
Dim atdr As AttachedDrawing
Dim amap As AcadMap
Dim boolVal As Boolean
Dim cDrset As Integer, i As Integer
```

```
Set amap = ThisDrawing.Application.
GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
Set drset = prj.DrawingSet
Set atdr = drset.Add("ALIASNAME:\\DirName\\DwgName1.dwg")
Set atdr = drset.Add("ALIASNAME:\\DirName\\DwgName2.dwg")
boolVal = drset.Remove("ALIASNAME:\\DirName\\DwgName1.dwg")
If boolVal = True Then
Debug.Print "Removed " & "DwgName1.dwg"
End If
cDrset = drset.Count
For i = 0 To cDrSet - 1
Set atdr = drset.Item(i)
If atdr.AliasPath = "ALIASNAME:\\DirName\\DwgName2.dwg" Then
drset.Remove (i)
Debug.Print "Removed " & "DwgName2.dwg"
End If
Next i
ThisDrawing.Application.ZoomExtents
```

DrawingSet.ZoomExtents method

Zooms to the extents of the attached drawings.

ZoomExtents() As Boolean

Returns True on success.

ZoomExtents zooms the display view, in or out, to match the combined extents of all the attached drawings in the drawing set. Extents define the smallest rectangle that encompasses all the objects in a drawing or set of drawings.

This one-line example assumes you altered queried objects of a drawing. For sample code for querying a drawing, click ■. Add the following line of code just before End Sub to display queried objects.

prj.DrawingSet.ZoomExtents

Use ThisDrawing.Application.ZoomExtents instead of prj.DrawingSet.ZoomExtents to display queried objects and restore the state of the command line, so you can enter commands.

LoomExtents method

See Also of the attached drawings.

s Boolean

cess.

the display view, in or out, to match the combined extents awings in the drawing set. Extents define the smallest passes all the objects in a drawing or set of drawings.

le assumes you altered queried objects of a drawing. For ying a drawing, click ■. Add the following line of code just splay queried objects.

nExtents

pplication.ZoomExtents instead of nExtents to display queried objects and restore the state of you can enter commands.
ErrorEntry.Add method

Adds a diagnostic parameter to the error.

Add(Source As EAdeClassId, _ Message As String, _ SQLStatement As String, _ Position As Long _) As DiagParam

Returns the added diagnostic parameter.

Source

Identifies the object that caused the error.

Message

A message string describing the error.

SQLStatement

The SQL string, if SQL processing caused the error.

Position

The position of the error in an SQL string.

The following example adds a diagnostic parameter to an error entry object.

Dim diapm As DiagParam

On Error GoTo ErrHandler ' Some error occurs while accessing the SQL database ' ... Exit Sub ErrHandler: Set diapm = amap.ErrorStack.Item(0).Add(122, "incorrect syntax", _ "SELECT * FROM RECORD3", 5)

ErrorEntry.Item method

Gets a diagnostic parameter associated with the error.

Item(Index As Long) As DiagParam

Returns the diagnostic parameter.

Index

Index, starting at 0, of a diagnostic parameter in the ErrorEntry collection.

The following example shows getting an error and then displaying the diagnostic parameters, message and SQL statement in the Immediate Window.

Dim diapm As DiagParam

On Error GoTo ErrHandler

```
' Some error occurs while accessing the SQL database
```

```
'...
Exit Sub
ErrHandler:
Set diapm = amap.ErrorStack.Item(0).Add(122, "incorrect syntax", _
"SELECT * FROM RECORD3", 5)
Debug.Print amap.ErrorStack.Item(0).Item(0).Message
Debug.Print amap.ErrorStack.Item(0).Item(0).SQLStatement
```

em method

<u>ee Also</u> cameter associated with the error.

ng) As DiagParam

ic parameter.

0, of a diagnostic parameter in the ErrorEntry collection.

ple shows getting an error and then displaying the diagnostic and SQL statement in the Immediate Window.

Param

andler *w*hile accessing the SQL database

rrorStack.Item(0).Add(122, "incorrect syntax", _ RECORD3", 5) rrorStack.Item(0).Item(0).Message rrorStack.Item(0).Item(0).SQLStatement

ErrorStack.Add method

Adds a user-defined error to the top of the error stack.

Add(ErrorCode As EErrCode, _ ErrorType As EAdeErrType, _ ErrorMessage As String _) As ErrorEntry

Returns the added ErrorEntry collection.

ErrorCode

An EErrCode constant representing an error that occurred.

ErrorType

An EAdeErrType constant identifying the type of error that occurred.

ErrorMessage

A string containing a description of the error.

The following example adds an error to the error stack, gets it from the stack, and displays it.

Dim amap As AcadMap Dim ee As ErrorEntry Set amap = ThisDrawing.Application. _ GetInterfaceObject("AutoCADMap.Application") Set ee = amap.ErrorStack. _ Add(345, kAdeWarning, "Abnormal termination") MsgBox "" & amap.ErrorStack.Item(0).ErrMessage

dd method

error to the top of the error stack. **As EErrCode**, _ .deErrType, _ .String _

rorEntry collection.

nstant representing an error that occurred.

e constant identifying the type of error that occurred.

ng a description of the error.

ple adds an error to the error stack, gets it from the stack,

Map
'y
ving.Application. _
'AutoCADMap.Application")
Stack. _
ing, "Abnormal termination")
ErrorStack.Item(0).ErrMessage

ErrorStack.Item method

Gets an error from the stack.

Item(Index As Long) As ErrorEntry

Returns the ErrorEntry collection.

Index

Index of the error, starting at 0, in the stack.

em method

<u>ee Also</u> ie stack.

ng) As ErrorEntry

ry collection.

r, starting at 0, in the stack.

ErrorStack.RemoveAll method

ErrorStack collection Example Empties the error stack.

RemoveAll() As Boolean

Returns True on success.

emoveAll method

<u>xample</u> Ck.

Boolean

cess.

Expression.Execute method

Expression object Executes the expression.

Execute(_ AcDbID as Long, _ ResultValue As Variant _) As Boolean

Returns True on success.

AcDbID

The target object of the expression, entered as a parameter of MapUtil.NewExpression.

If the argument is 0, the expression is evaluated without using an object; otherwise, the expression is evaluated with the object identified by the AcDbObjectID. Expressions fail if the target object is illogical. For example, if you add the expression (".AREA") to a report template, the expression reports the area of objects if the objects have area. When the expression executes on target objects without area, no output is sent to the report. The report on two objects, a stream without an area and a water body with an area, that uses a template with expressions, ".LENGTH", ".LAYER", and ".AREA" looks like this:

362,STREAM,

2496,WATER,459170

ResultValue

The result of the execution of the expression.

The following example.creates and executes an expression that finds the area of all objects in model space.

Dim amap As AcadMap Dim prj As Project Dim exp As Expression Dim i As Integer Dim varVal As Variant Dim obj As Object Dim lngAry() As Long ReDim lngAry(ThisDrawing.ModelSpace.Count)

Dim strVal As String

```
Set amap = ThisDrawing.Application. _
GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
Set exp = prj.MapUtil.NewExpression(".AREA")
For Each obj In ThisDrawing.ModelSpace
IngAry(i) = obj.ObjectID
exp.Execute IngAry(i), varVal
i = i + 1
strVal = strVal & varVal & Chr(13)
Next
MsgBox strVal
```

FenceBound.Add method

<u>FenceBound collection</u> Adds a point to a FenceBound collection.

Add(X As Double, Y As Double, Z As Double) As Point3d

Returns a Point3d object.

X, Y, Z

Coordinates of the point to add.

The Z value is ignored.

Add method

nceBound collection.

e, Y As Double, Z As Double) As Point3d

oject.

ie point to add. ed.

FenceBound.Item method

<u>FenceBound collection</u> Gets a point from a FenceBound collection.

Item(PointID As Long) As Point3d

Returns a Point3d object.

PointID

The index of the point to get, starting at 0.

Item method

FenceBound collection.

Long) As Point3d

oject.

point to get, starting at 0.

FenceBound.Remove method

<u>FenceBound collection</u> Removes a point from a FenceBound collection.

Remove(PointID As Long) As Boolean

Returns True on success.

PointID

The index of the point to remove, starting at 0.

Remove method

m a FenceBound collection.

As Long) As Boolean

cess.

point to remove, starting at 0.

MapUtil.NewBufferFence method

Creates a buffer fence boundary.

NewBufferFence() As BufferFenceBound

Returns a BufferFenceBound collection.

⁷BufferFence method

boundary.

) As BufferFenceBound

ceBound collection.

MapUtil.NewBufferPolyline method

Creates a buffer polyline boundary.

NewBufferPolyline(_ ObjId As Long, _ Width As Double _) As BufferPolylineBound

Returns a BufferPolylineBound collection.

ObjId

ID of a polyline object that will define the new buffer polyline boundary.

Width

Buffer width.

⁷BufferPolyline method

^{jo} Tine boundary.

ıe(_

-

e_____ neBound

/lineBound collection.

object that will define the new buffer polyline boundary.

MapUtil.NewCircle method

Creates a circle boundary.

NewCircle(_ Radius As Double, _ X As Double, _ Y As Double _) As CircleBound

Returns a CircleBound object.

Radius

Radius

Χ, Υ

Center coordinates

⁷Circle method

idary.

e, _

l

nd object.

es
MapUtil.NewClosedPolyline method

Creates a closed polyline boundary.

NewClosedPolyline(ObjId As Long) As ClosedPolylineBound

Returns a ClosedPolylineBound collection.

ObjId

The ID of a polyline object that will define the new closed polyline boundary.

⁷ClosedPolyline method

⁹/line boundary. ne(ObjId As Long) As ClosedPolylineBound ylineBound collection.

line object that will define the new closed polyline

MapUtil.NewExpression method

<u>MapUtil object</u> <u>See Also</u> Creates an Expression object.

NewExpression(Expression As String) As Expression

Returns an Expression object.

Expression

The expression string.

You can use the Expression object to specify a property alteration or a query report, or use it to execute general-purpose expressions, such as an expression to get the area of all objects in a selection set. You can use the dot, colon, ampersand, SQL, and AutoLISP variables as expression strings. For more information on expressions, see AutoCAD Map UI Help, "Expressions".

The following example creates an Expression object that specifies the area of a drawing object.

Dim amap As AcadMap Dim prj As Project Dim exp As Expression Dim i As Integer Dim varVal As Variant Dim obj As Object Dim lngAry() As Long ReDim lngAry(ThisDrawing.ModelSpace.Count) Dim strVal As String

Set amap = ThisDrawing.Application. _ GetInterfaceObject("AutoCADMap.Application") Set prj = amap.Projects(ThisDrawing) Set exp = prj.MapUtil.NewExpression(".AREA")

MapUtil.NewFence method

Creates a new fence boundary.

NewFence() As FenceBound

Returns a FenceBound collection.

⁷Fence method

¹⁰boundary. **CenceBound** nd collection.

MapUtil.NewLocationAll method

Creates an unlimited boundary.

NewLocationAll() As AllBound

Returns the new AllBound object.

The following example calls MapUtil.NewLocationAll to specify selecting all objects in a query. Before running this code, attach a drawing.

Dim amap As AcadMap Dim prj As Project Dim qry As Query Dim mainqrybr As QueryBranch Dim qrylf As QueryLeaf Dim boolVal As Boolean

```
Set amap = ThisDrawing.Application. _
GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
Set qry = prj.CurrQuery
Set mainqrybr = qry.QueryBranch
Set qrylf = mainqrybr.Add(kLocationCondition, kOperatorAnd)
boolVal = qrylf.SetLocationCond(kLocationInside,
prj.MapUtil.NewLocationAll)
qry.Mode = kQueryDraw
boolVal = qry.Define(mainqrybr)
boolVal = qry.Execute
ThisDrawing.Application.ZoomExtents
```

⁷LocationAll method

boundary.

) As AllBound

3ound object.

ple calls MapUtil.NewLocationAll to specify selecting all efore running this code, attach a drawing.

∕Іар

\ueryBranch Leaf lean wing.Application. __ 'AutoCADMap.Application'') cts(ThisDrawing) ery QueryBranch r.Add(kLocationCondition, kOperatorAnd) ocationCond(kLocationInside, :ationAll) Draw !(mainqrybr) te ation.ZoomExtents

MapUtil.NewODFieldDefs method

MapUtil object Example See Also Creates a field definition collection.

NewODFieldDefs() As ODFieldDefs

Returns an ODFieldDefs collection.

ODFieldDefs collections contain field definitions for object data tables.

The following example creates field definitions for an object data table.

Dim amap As AcadMap Dim ODfdfs As ODFieldDefs

Set amap = ThisDrawing.Application. _ GetInterfaceObject("AutoCADMap.Application") Set ODfdfs = amap.Projects(ThisDrawing).MapUtil.NewODFieldDefs

⁷ODFieldDefs method

<u>le</u> <u>See Also</u> tion collection.

() As ODFieldDefs

Defs collection.

ions contain field definitions for object data tables.

ple creates field definitions for an object data table.

∕Iap FieldDefs

wing.Application. _
'AutoCADMap.Application")
Projects(ThisDrawing).MapUtil.NewODFieldDefs

MapUtil.NewPoint3d method

Creates a point boundary.

NewPoint3d(_ X As Double, _ Y As Double, _ Z As Double _) As Point3d

Returns a Point3d object.

X, Y, Z

Coordinates

⁷Point3d method

<u>le See Also</u> dary.

)ject.

MapUtil.NewPolygon method

Creates a polygon boundary.

NewPolygon() As PolygonBound Returns a PolygonBound collection. The new collection is empty. Add points to it.

⁷Polygon method

⁵⁰ **s PolygonBound** ound collection. s empty. Add points to it.

MapUtil.NewPolyline method

Creates a polyline boundary.

NewPolyline(ObjId As Long) As PolylineBound

Returns a PolylineBound collection.

ObjId

The ID of a polyline object that will define the new polyline boundary.

⁷Polyline method

⁵⁰ Jundary. Id As Long) As PolylineBound ound collection.

line object that will define the new polyline boundary.

MapUtil.NewWindow method

Creates a window boundary.

```
NewWindow( _
BottomLeft As Point3d, _
UpperRight As Point3d _
) As WindowBound
```

Returns a WindowBound object.

BottomLeft

The bottom left coordinate point of a new window.

UpperRight

The upper right coordinate point of a new window

This example is an excerpt from the sample code for creating a query. For more information, click . The following example gets the extents of the drawing to define a query window boundary.

Dim wind As WindowBound Dim dblary As Variant

```
'Get DWG Extents
dblary = prj.DrawingSet.Item("MAPTUT:\\citymap7.dwg").Extents
Set mapu = map.Projects(ThisDrawing).MapUtil
Set wind = mapu.NewWindow( _
mapu.NewPoint3d(dblary(0), dblary(1), 0), _
mapu.NewPoint3d(dblary(2), dblary(3), 0))
```

Window method

<u>le <u>See Also</u> Jundary.</u>

oint3d, _ oint3d _ nd ound object.

coordinate point of a new window.

coordinate point of a new window

xcerpt from the sample code for creating a query. For more The following example gets the extents of the drawing to w boundary.

wBound nt

\$Set.Item("MAPTUT:\\citymap7.dwg").Extents
jects(ThisDrawing).MapUtil
wWindow(_
blary(0), dblary(1), 0), _
blary(2), dblary(3), 0))

NestedDrawings.Item method

NestedDrawings collection

Gets an object reference to a drawing that is part of a NestedDrawings collection.

Item(DwgNameOrIndex As Variant) As AttachedDrawing

Returns a reference to a specified AttachedDrawing object that belongs to NestedDrawings.

DwgNameOrIndex

Contains the full name of the drawing file (alias + path + filename) or the index position, starting at 0, of the AttachedDrawing object within the NestedDrawings collection.

The following example gets a nested drawing.

Dim amobj as AcadMap Dim pobj As AutoCADMap.Project Dim dsobj as DrawingSet Dim adobj as AttachedDrawing Dim ndobj as NestedDrawing Dim objnum as Long Set amobj = acadApp.GetInterfaceObject("AutoCADMap.Application") Set pobj = amobj.Projects(ThisDrawing) Set dsobj = pobj.DrawingSet Set ndobj = dsobj.NestedDrawings Set adobj = ndobj.Item(5)

ngs.Item method

<u>on</u>

nce to a drawing that is part of a NestedDrawings

rIndex As Variant) As AttachedDrawing

o a specified AttachedDrawing object that belongs to

X

name of the drawing file (alias + path + filename) or the arting at 0, of the AttachedDrawing object within the collection.

ple gets a nested drawing.

ODFieldDefs.Add method

<u>ODFieldDefs collection</u> <u>Example</u> <u>See Also</u> Adds a field definition.

Add(_ FieldName As String, _ FieldDescription As String, _ DefaultValue As Variant, _ Index As Long _) As ODFieldDef

Returns a copy of the field you added.

FieldName

Field name.

FieldDescription

Field description.

DefaultValue

Default field value.

Index

The index, starting at 0, where the field is inserted. Existing fields at that index and higher move up.

.Add method

Example See Also **Dn.**

ring, _ As String, _ Variant, _

e field you added.

ıe.

•

ng at 0, where the field is inserted. Existing fields at that move up.

ODFieldDefs.Item method

ODFieldDefs collection Gets a field from the collection.

Item(Index As Variant) As ODFieldDefs

Returns the specified field.

Index

Field name or index, starting at 0.

.Item method

collection. riant) As ODFieldDefs l field.

lex, starting at 0.

ODFieldDefs.Remove method

<u>ODFieldDefs collection</u> Removes an object data field from the collection.

Remove(Index As Variant) As Boolean

Returns True on success.

Index

Name or index, starting at 0, of the field to remove.

Note You cannot remove the last field.
.Remove method

lata field from the collection.

: Variant) As Boolean cess.

tarting at 0, of the field to remove. nove the last field.

ODRecord.AttachTo method

<u>ODRecord object</u> <u>Example</u> <u>See Also</u> Attaches a record to a drawing object.

AttachTo(DbID As Long) As Boolean

Returns True on success. If False, check the error stack.

DbID

The ID of the object to which this record will be attached.

ttachTo method

<u>a drawing object.</u>

.s Long) As Boolean

cess. If False, check the error stack.

ject to which this record will be attached.

ODRecord.Item method

Gets a field. See Also

Item(Index As Variant) As ODFieldValue

Returns the field.

Index

The index, starting at 0, of the field to get, or the field's name.

em method

<u>Also</u>

riant) As ODFieldValue

ıg at 0, of the field to get, or the field's name.

ODRecords.Init method

<u>ODRecords collection</u> <u>See Also</u> Initializes an ODRecords collection with the records of a given drawing object.

Init(_ acadObject As Object, _ OpenMode As Boolean, _ SkipNested As Boolean _) As Boolean

Returns True on success.

acadObject

The drawing object whose records you will read or edit.

OpenMode

If True, records are read-write. If False, they are read-only.

SkipNested

True inhibits scanning of records attached to subordinate objects.

You can instantiate an ODRecords collection by calling either ODTable.GetODRecords() or ODTables.GetODREcords(). In the first case, you are referencing a particular object-data table definition. In the second, you are referencing all object-data table definitions in the project. When you initialize the ODRecords collection in the first case, you initialize it with all the records attached to the given object that match a particular table definition. When you initialize the ODRecords collection in the second case, you in itialize it with all the records attached to the given object that match any table definition.

Note You must explicitly release an ODRecords collection when you are finished with it.

The following sample demonstrates reading and editing existing records. The ODRecords collection is created before we enter the loop, and it is not released until we exit the loop, because each call to Init() reuses the same collection object. The sample assumes a selection set, ss, and an object-data table, oTable2.

```
Set oRecords = oTable2.Get0DRecords()
Dim oDrawingObject As AcadEntity
Dim bRetVal As Boolean
For Each oDrawingObject In ss
    bRetVal = oRecords.Init(oDrawingObject, True, True)
    While oRecords.IsDone = False
```

```
oRecord.Item(1).Value = oRecord.Item(0).Value
    oRecords.Next
    Wend
Next
Set oRecords = Nothing
```

Note Although you can read or edit records, you cannot add or remove records during the life of an ODRecords object.

The following sample demonstrates iterating existing records and then adding records based on what we have found. In this situation, the ODRecords collection is created within the loop and then released within the loop before any records are added. The sample assumes a selection set, ss, and an object-data table, oTable2.

```
Dim oDrawingObject As AcadEntity
Dim bRetVal As Boolean
Dim bHasRecords As Boolean
Dim oRecord As ODRecord
For Each oDrawingObject In ss
  bHasRecords = False
  Set oRecords = oTable2.GetODRecords()
  bRetVal = oRecords.Init(oDrawingObject, True, True)
  While oRecords.IsDone = False
      HasRecords = True
      oRecords.Next
  Wend
  Set oRecords = Nothing
   If HasRecords = False Then
      Set oRecord = oTable2.CreateRecord
      oRecord.Item(0).Value = "0"
      bRetVal = oRecord.AttachTo(CLng(oDrawingObject.ObjectID))
  End If
Next
Set oRecord = Nothing
```

ODRecords.Next method

ODRecords collection See Also Makes the next record current.

Next() As Boolean

Returns True on success.

Create an object data table called SampleOD. For more information, click **2**. Run this code to step through the records of each object in the drawing using ODRecords.Next.

Dim amap As AcadMap Dim ODrcs As ODRecords Dim boolVal As Boolean Dim acadObj As Object Dim prj As Project Dim i As Integer

Set amap = ThisDrawing.Application. GetInterfaceObject("AutoCADMap.Application") Set prj = amap.Projects(ThisDrawing) prj.ProjectOptions.DontAddObjectsToSaveSet = True Set ODrcs = prj.ODTables.GetODRecords For Each acadObj In ThisDrawing.ModelSpace boolVal = ODrcs.Init(acadObj, True, False) Do While ODrcs.IsDone = False Debug.Print ODrcs.Record.tableName Debug.Print ODrcs.Record.ObjectID For i = 0 To ODrcs.Record.Count - 1 Debug.Print ODrcs.Record.Item(i).Value Next i **ODrcs.Next** Loop Next

ODRecords.Record method

Gets the current record.

Record() As ODRecord

Returns the record.

Record method

<u>See Also</u> r**d.**

Record

ODRecords.Remove method

Removes the current record from the collection.

Remove() As Boolean

Returns True on success.

The record is detached from its drawing object.

Create the query described here **D**. Create an object data table described here **D**. Run the following code to remove all the records with fields called Streams and display a list of the remaining records.

Dim amap As AcadMap Dim ODrcs As ODRecords Dim boolVal As Boolean Dim acadObj As Object Dim prj As Project Dim i As Integer

```
Set amap = ThisDrawing.Application. _
GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
prj.ProjectOptions.DontAddObjectsToSaveSet = True
Set ODrcs = prj.ODTables.GetODRecords
For Each acadObj In ThisDrawing.ModelSpace
boolVal = ODrcs.Init(acadObj, True, False)
Do While ODrcs.IsDone = False
For i = 0 To ODrcs.Record.Count - 1
If ODrcs.Record.Item(i).Value = "STREAM" Then
ODrcs.Remove
End If
Next i
ODrcs.Next
Loop
Next
ODrcs.Rewind
For Each acadObj In ThisDrawing.ModelSpace
boolVal = ODrcs.Init(acadObj, True, False)
Do While ODrcs.IsDone = False
Debug.Print ODrcs.Record.tableName
```

Debug.Print ODrcs.Record.ObjectID For i = 0 To ODrcs.Record.Count - 1 Debug.Print ODrcs.Record.Item(i).Value Next i ODrcs.Next Loop Next

ODRecords.Rewind method

ODRecords collection See Also Makes the first record current.

Rewind() As Boolean

Returns True on success.

Rewind method

<u>See Also</u> d current.

lean

cess.

ODRecords.Update method

Replaces the current record.

Update(ODRecord As ODRecord) As Boolean

Returns True on success.

ODRecord

The record that replaces the current record.

Jpdate method

<u>See Also</u> record.

d As ODRecord) As Boolean

cess.

eplaces the current record.

ODTable.CreateRecord method

<u>ODTable object</u> <u>Example</u> <u>See Also</u> Creates a record.

CreateRecord() As ODRecord

Returns the new empty record.

ateRecord method

<u>ole</u> <u>See Also</u>

s ODRecord

ty record.

ODTable.GetODRecords method

Gets the collection of all records in the table.

GetODRecords() As ODRecords

Returns the collection.

Note that the ODRecords collection iterates the records of only one object at a time, as set by ODRecords.Init. If you get ODRecords through an ODTables collection, you can iterate the object records regardless of which table they are from. If you get ODRecords through an ODTable object, you can iterate only those object records from that table, as shown in the following example.

The following example builds on sample code for creating object data tables. For more information, click . The example shows how to get the records of a selected object and print them in the Immediate Window. Add the following code to the end of the sample code for creating object data tables. Change the name of the table from SampleOD to another name and run the sample again.

Dim ODrcs As ODRecords Dim boolVal As Boolean Dim returnObj As AcadObject Dim basePnt As Variant Dim i As Integer

Set ODrcs = amap.Projects.Item(ThisDrawing). _ ODTables.Item("SampleOD").GetODRecords

'Prompt user to select an object ThisDrawing.Utility.GetEntity returnObj, basePnt, "Select an object" boolVal = ODrcs.Init(returnObj, True, False) Debug.Print ODrcs.Record.tableName Debug.Print ODrcs.Record.ObjectID For i = 0 To ODrcs.Record.Count - 1 Debug.Print ODrcs.Record.Item(i).Value Next i

ODTables.Add method

Creates an object data table.

Add(_ TableName As String, _ TableDescription As String, _ ODFieldDefs As ODFieldDefs, _ StoreAsXData As Boolean _) As ODTable

Returns the specified table.

TableName

At most 29 characters, no spaces.

TableDescription

The description of the table.

ODFieldDefs

Field definitions for the table.

StoreAsXData

True: Record data is stored with drawing objects as extended entity data.

False: Record data is stored with drawing objects as AutoCAD Xrecords.

The following code is an excerpt from a larger example. The code shows how to prevent an error by checking that the table you want to add doesn't already exist. For the entire example, click **B**.

Dim amap As AcadMap Dim ODfdfs As ODFieldDefs Dim ODtb As ODTable

'Ensure Table Does Not Exist
If amap.Projects(ThisDrawing) _
.ODTables.Item("TableA") Is Nothing Then

'Register OD Table in the drawing
Set ODtb = amap.Projects(ThisDrawing) _
.ODTables.Add("TableA", "Sample Xdata", ODfdfs, True)

. . . End If

ODTables.GetODRecords method

Gets the collection of all records in the project.

GetODRecords() As ODRecords

Returns the collection.

Includes all the records in all the tables in the project. Note that the ODRecords collection iterates the records of only one object at a time, as set by ODRecords.Init. If you get ODRecords through an ODTables collection, you can iterate the object records regardless of which table they are from, as shown in the following example. If you get ODRecords through an ODTable object, you can iterate only those object records from that table.

The following example builds on sample code for creating object data tables. For more information, click . First, run the sample code for creating object data tables twice. Change the name of the table from SampleOD to SampleOD2 before running it the second time. Next, run the following code to iterate through records from multiple tables attached to each object in the drawing.

Dim amap As AcadMap Dim ODrcs As ODRecords Dim boolVal As Boolean Dim acadObj As Object Dim prj As Project Dim i As Integer

```
Set amap = ThisDrawing.Application. _
GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
prj.ProjectOptions.DontAddObjectsToSaveSet = True
Set ODrcs = prj.ODTables.GetODRecords
For Each acadObj In ThisDrawing.ModelSpace
boolVal = ODrcs.Init(acadObj, True, False)
Do While ODrcs.IsDone = False
Debug.Print ODrcs.Record.tableName
Debug.Print ODrcs.Record.ObjectID
For i = 0 To ODrcs.Record.Count - 1
Debug.Print ODrcs.Record.Item(i).Value
Next i
ODrcs.Next
```

Loop Next
ODTables.Item method

Gets a table. See Also

Item(Index As Variant) As ODTable

Returns the specified table.

Index

The table name or index, starting at 0.

The following line of code, an excerpt from the ODTable.GetODRecords example, shows how to use the ODTables.Item method to get the records of the table.

Dim ODrcs As ODRecords Set ODrcs = amap.Projects.Item(ThisDrawing). _ ODTables.Item("SampleOD").GetODRecords

m method

<u>e Also</u>

riant) As ODTable

l table.

r index, starting at 0.

f code, an excerpt from the ODTable.GetODRecords ' to use the ODTables.Item method to get the records of the

.ecords rojects.Item(ThisDrawing). _ npleOD'').GetODRecords

PolygonBound.Add method

<u>PolygonBound collection</u> Adds a point to a PolygonBound collection.

Add(X As Double, _ Y As Double, _ Z As Double _) As Point3d

Returns a Point3d object.

X, Y, Z

Coordinates of the point to add.

The Z value is ignored.

ıd.Add method

lygonBound collection.

e, _

)ject.

e point to add.

ed.

PolygonBound.Item method

<u>PolygonBound collection</u> Gets a point from a PolygonBound collection.

Item(PointID As Long) As Point3d

Returns a Point3d object.

PointID

The index of the point to get, starting at 0.

ıd.Item method

PolygonBound collection.
Long) As Point3d
oject.

point to get, starting at 0.

PolygonBound.Remove method

<u>PolygonBound collection</u> Removes a point from a PolygonBound collection.

Remove(PointID As Long) As Boolean

Returns True on success.

PointID

The index of the point to remove, starting at 0.

ıd.Remove method

¹ m a PolygonBound collection. As Long) As Boolean

cess.

point to remove, starting at 0.

PolylineBound.Item method

<u>PolylineBound collection</u> Gets a point from a PolylineBound object.

Item(PointID As Long) As Point3d

Returns a Point3d object.

PointID

The index of the point to get, starting at 0.

ıd.Item method

PolylineBound object.
Long) As Point3d
oject.

point to get, starting at 0.

Project.RunExternalQuery method

Loads an external query; if it's an auto-execute query, it runs.

RunExternalQuery(PathToExternalQuery As String) As Boolean

Returns False if the query was not found or did not run.

PathToExternalQuery

Alias path and file name of the external query.

The query runs when it is loaded if was saved as an auto-execute query. To save an auto-execute query, include the ESaveQueryOptions constant, kAutoExecute, in the SaveOptions argument when you call Query.Save.

xternalQuery method

ery; if it's an auto-execute query, it runs.

y(PathToExternalQuery As String) As Boolean

juery was not found or did not run.

ıery

le name of the external query.

i it is loaded if was saved as an auto-execute query. To save y, include the ESaveQueryOptions constant, kAutoExecute, rgument when you call Query.Save.

Project.WhoLocksObject method

Gets information about a locked drawing object.

WhoLocksObject(_ EntID As Long, _ ProjectName As String, _ DrawingName as String, _ OwnerName As String, _ Date As String, _ Time As String _) As Boolean

Returns False if the object is not locked.

EntID

The AcDbObjectID identifier of the drawing object.

ProjectName (Output)

Name of the document in the project locking the object.

DrawingName (Output)

Name of the drawing to which the object EntID belongs.

OwnerName (Output)

Login name of the user who locked the object.

Date(Output)

Date the object was locked.

Time(Output)

Time the object was locked.

Projects.Item method

Gets an AutoCAD Map project.

Item(Index As Variant) As Project

Returns the project.

Index

Index, starting at 0, of a project in the collection or an AutoCAD document object.

You get the AutoCAD document from the AcadDocuments collection, and use it to index into the Projects collection.

method

lap project. riant) As Project

0, of a project in the collection or an AutoCAD document

D document from the AcadDocuments collection, and use it jects collection.

Query.Clear method

<u>Query object</u> <u>Example</u> Clears the query of all but the main branch and mode.

Clear() As Boolean

Returns True on success.

Call this method before setting a new query. The following example clears a query definition. Before running this code, attach a drawing. Dim amap As AcadMap

Dim prj As Project Dim qry As Query Dim mainqrybr As QueryBranch Dim qrylf As QueryLeaf Dim boolVal As Boolean

Set amap = ThisDrawing.Application. _ GetInterfaceObject("AutoCADMap.Application") Set prj = amap.Projects(ThisDrawing) Set qry = prj.CurrQuery qry.Clear

nethod

ll but the main branch and mode.

an

cess.

ore setting a new query. The following example clears a fore running this code, attach a drawing. Map

}ueryBranch Leaf →lean

```
wing.Application. _
'AutoCADMap.Application")
cts(ThisDrawing)
ery
```

Query.Define method

<u>Query object</u> <u>Example</u> <u>See Also</u> Sets or replaces the query's main branch.

Define(QueryBranch As QueryBranch) As Boolean

Returns True on success.

QueryBranch

The main branch of a query.

Note To prevent an error, call Query.Define before attempting to use Query.StringContext.

The following example shows the minimal code for setting up and defining a query. Before running this code, attach a drawing.

Dim amap As AcadMap Dim prj As Project Dim qry As Query Dim qrybr As QueryBranch Dim qrylf As QueryLeaf Dim boolVal As Boolean

Set amap = ThisDrawing.Application. __ GetInterfaceObject("AutoCADMap.Application") Set prj = amap.Projects(ThisDrawing) Set qry = prj.CurrQuery Set qrybr = qry.QueryBranch Set qrylf = qrybr.Add(kLocationCondition, kOperatorAnd) boolVal = qrylf.SetLocationCond(kLocationInside, __ prj.MapUtil.NewLocationAll) qry.Mode = kQueryDraw boolVal = qry.Define(qrybr)

Query.Execute method

<u>Query object</u> <u>Example</u> <u>See Also</u> Executes the current query.

Execute As Boolean

Returns True on success.

The following example executes a query and calls the Application.ZoomExtents to display the query results. Before running this code, attach a drawing.

Dim amap As AcadMap Dim prj As Project Dim qry As Query Dim mainqrybr As QueryBranch Dim qrylf As QueryLeaf Dim boolVal As Boolean

```
Set amap = ThisDrawing.Application. _
GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
Set qry = prj.CurrQuery
qry.Clear
Set mainqrybr = qry.QueryBranch
Set qrylf = mainqrybr.Add(kLocationCondition, kOperatorAnd)
boolVal = qrylf.SetLocationCond(kLocationInside, _
prj.MapUtil.NewLocationAll)
qry.Mode = kQueryDraw
boolVal = qry.Define(mainqrybr)
boolVal = qry.Execute
ThisDrawing.Application.ZoomExtents
```

e method

<u>See Also</u> query.

an

cess.

ple executes a query and calls the Application.ZoomExtents results. Before running this code, attach a drawing.

∕Лар

)ueryBranch Leaf)lean

```
wing.Application. _
'AutoCADMap.Application")
cts(ThisDrawing)
ery
```

```
QueryBranch
r.Add(kLocationCondition, kOperatorAnd)
ocationCond(kLocationInside, _
:ationAll)
Draw
!(mainqrybr)
te
ation.ZoomExtents
```

Query.Load method

Gets a query from the query library.

Load(SavedQuery As Query) As Boolean

Returns True on success.

SavedQuery

Represents the query taken from the query library.

The query is made current. Note that this method does not load a query from an external file.

The following example shows how to set a saved query to the member of the QueryCategories collection you want to load, call Load, and execute the loaded query. Before running this code, attach a drawing, and save a query in the query library by running the Query.Save example. To go to Query.Save, click **P**.

Dim amap As AcadMap Dim prj As Project Dim qry As Query Dim mainqrybr As QueryBranch Dim qrylf As QueryLeaf Dim boolVal As Boolean Dim i As Integer, cqryct As Integer Dim j As Integer, cqry As Integer Dim sqry As SavedQuery

```
Set amap = ThisDrawing.Application. _
GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
prj.CurrQuery.Clear
For i = 0 To prj.QueryCategories.Count - 1
If prj.QueryCategories(i).Name = "DrawingCategory" Then
cqryct = i
End If
Next i
For j = 0 To prj.QueryCategories.Item(cqryct).Count - 1
If prj.QueryCategories.Item(cqryct).Count - 1
If prj.QueryCategories.Item(cqryct).Item(j).Name = "myInternalquery" Then
cqry = j
End If
```

Next j Set sqry = prj.QueryCategories.Item(cqryct).Item(cqry) boolVal = prj.CurrQuery.Load(sqry) boolVal = prj.CurrQuery.Execute ThisDrawing.Application.ZoomExtents

Query.Save method

Saves the current query to the query library or an external file.

Save(External As Boolean, ______ kSaveOptions As ESaveQueryOptions, __ Category As String, ___ Name As String, ____ Description As String, ____ Filename as String _____) As Boolean

Returns True on success.

External

True saves the query to the external file named Filename. False saves the query to the Query Library.

kSaveOptions

Specifies information to save externally.

Set the External property to True before using kSaveOptions.

You combine kSaveOptions with the + operator. For example, to save the coordinates of the query and the alter properties, enter kQrySaveCoordinates + kQrySaveAlteration.

Category

The name of the category to which the query is added.

Name

A unique name for the query.

Description

The query description.

Filename

The full path of the external file for saving the query.

To save a query to an external file, set External to True and specify the file name. Use a second backslash to delimit each backslash in the path.

The following example first saves a query with a description and a category

name in an external file and then also saves it in the query library. You pass an empty string, as shown in the second call to Query.Save, instead of a file name to save the query in the library.

Dim amap As AcadMap Dim prj As Project Dim qry As Query Dim mainqrybr As QueryBranch Dim qrylf As QueryLeaf Dim boolVal As Boolean

```
Set amap = ThisDrawing.Application.
GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
Set qry = prj.CurrQuery
qry.Clear
Set mainqrybr = qry.QueryBranch
Set grylf = maingrybr.Add(kLocationCondition, kOperatorAnd)
boolVal = qrylf.SetLocationCond(kLocationInside, _
prj.MapUtil.NewLocationAll)
qry.Mode = kQueryDraw
boolVal = qry.Define(mainqrybr)
'Only if C:\\query.qry does not exist continue -- code not shown
boolVal = qry.Save( _
True, 0, "DrawingCategory", "myExternalquery", ____
"Qry on disk", "C:\\query.qry")
boolVal = qry.Save( __
False, 0, "DrawingCategory", "myInternalquery", _
"Qry in library", "")
```
QueryBranch.Add method

<u>QueryBranch collection</u> <u>Example</u> <u>See Also</u> Adds a leaf or sub-branch to a query.

Add(QueryBranchOrLeaf As EClassID, _ JoinOp As EJoinOperator) As Object

Returns a QueryBranch or QueryLeaf object.

QueryBranchOrLeaf

Type of leaf (condition) or sub-branch you are adding.

JoinOp

Join operator

The following EClassID values are valid QueryBranchOrLeaf arguments.

kQueryUnit kQueryCondition kQueryBranch kPropertyCondition kDataCondition kLocationCondition kSQLCondition

The following example adds a sub-branch to a query.

Dim amap As AcadMap Dim prj As Project Dim qry As Query Dim mainqrybr As QueryBranch Dim qrylf As QueryLeaf Dim andqrybr As QueryBranch

Set amap = ThisDrawing.Application. _ GetInterfaceObject("AutoCADMap.Application") Set prj = amap.Projects(ThisDrawing) Set qry = prj.CurrQuery qry.Clear Set mainqrybr = qry.QueryBranch Set qrylf = mainqrybr.Add(kLocationCondition, kOperatorAnd) Set andqrybr = mainqrybr.Add(kQueryBranch, kOperatorAnd)

QueryBranch.Clear method

<u>QueryBranch collection</u> <u>See Also</u> Deletes all branches and leaves from this branch downward.

Clear() As Boolean

Returns nothing.

Consider the sample, <u>Creating a Query</u>. Add the following line to the sample before qry.Define:

andqrybr.Clear

Run the revised sample and then look at the query in the AutoCAD Map UI. Notice that only the Location condition is defined, because andqrybr.Clear cleared the data and property conditions.

.Clear method

See Also and leaves from this branch downward.

an

, <u>Creating a Query</u>. Add the following line to the sample

ple and then look at the query in the AutoCAD Map UI. Location condition is defined, because andqrybr.Clear property conditions.

QueryBranch.Item method

Returns a sub-branch or query leaf from a branch.

Item(NumObject As Long) As QueryBranch

Returns the specified branch or leaf.

NumObject

Index of a sub-branch or leaf in a branch, starting at 0.

The following example builds on sample code for querying objects. For more information, click **>**. Add the following code to the example after qry.Define.This example uses two for loops to get all the branches and leaves and print their types.

```
' Define query before running this code
For i = 0 To mainqrybr.Count - 1
Debug.Print mainqrybr.Type
Debug.Print mainqrybr.Item(i).Type
Next i
For i = 0 To andqrybr.Count - 1
Debug.Print andqrybr.Type
Debug.Print andqrybr.Item(i).Type
Next i
```

.Item method

See Also or query leaf from a branch.
As Long) As QueryBranch
I branch or leaf.

anch or leaf in a branch, starting at 0.

ple builds on sample code for querying objects. For more Add the following code to the example after nple uses two for loops to get all the branches and leaves

e running this code 'br.Count - 1 'br.Type 'br.Item(i).Type

r.Count - 1 r.Type r.Item(i).Type

QueryBranch.Remove method

<u>QueryBranch collection</u> <u>See Also</u> Deletes a sub-branch or a leaf from a branch.

Remove(NumObject As Long) As Boolean

Returns True on success.

NumObject

Index of a branch or leaf, starting at 0.

The following example builds on sample code for querying objects. For more information, click . Add the following code to the example after qry.Define.This example prints the starting number of items in the.sub-branch collection called andqrybr, searches the collection for the kDataCondition leaf, removes it, and checks that the number of items in the sub-branch reflects the removal.

```
For i = 0 To andqrybr.Count - 1
Debug.Print andqrybr.Count
If andqrybr.Item(i).Type = kDataCondition Then
andqrybr.Remove (i)
End If
Next i
Debug.Print andqrybr.Count
```

.Remove method

```
    See Also
or a leaf from a branch.
    ect As Long) As Boolean
    cess.
```

ı or leaf, starting at 0.

ple builds on sample code for querying objects. For more Add the following code to the example after nple prints the starting number of items in the.sub-branch qrybr, searches the collection for the kDataCondition leaf, ks that the number of items in the sub-branch reflects the

r.Count - 1 vr.Count ype = kDataCondition Then

or.Count

QueryCategories.Add method

<u>QueryCategories collection</u> Creates a query category

Add(CategoryName As String) As QueryCategory

Returns the query category.

CategoryName

Name of the category enclosed in quotes.

ries.Add method

on gory ne As String) As QueryCategory tegory.

gory enclosed in quotes.

QueryCategories.Item method

<u>QueryCategories collection</u> Gets a query category from the library.

Item(Index As Variant) As QueryCategory

Returns the query category.

Index

Name or index, starting at 0, of the category to get

The following example shows how to set a saved query to the member of the QueryCategories collection you want to load, call Load, and execute the loaded query. Before running this code, attach a drawing, and save a query in the query library by running the Query.Save example. To go to Query.Save, click **2**.

```
Dim amap As AcadMap
Dim prj As Project
Dim qry As Query
Dim maingrybr As QueryBranch
Dim qrylf As QueryLeaf
Dim boolVal As Boolean
Dim i As Integer, cqryct As Integer
Dim j As Integer, cqry As Integer
Dim sqry As SavedQuery
Set amap = ThisDrawing.Application.
GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
prj.CurrQuery.Clear
For i = 0 To prj.QueryCategories.Count - 1
If prj.QueryCategories(i).Name = "DrawingCategory" Then
cqryct = i
End If
Next i
For j = 0 To prj.QueryCategories.Item(cgryct).Count - 1
If prj.QueryCategories.Item(cgryct).Item(j).Name = "myInternalquery" Then
cqry = j
End If
Next j
```

```
Set sqry = prj.QueryCategories.Item(cqryct).Item(cqry)
```

boolVal = prj.CurrQuery.Load(sqry)
boolVal = prj.CurrQuery.Execute
ThisDrawing.Application.ZoomExtents

QueryCategories.Remove method

<u>QueryCategories collection</u> Removes a query category from the library.

Remove(Index As Variant) As Boolean

Returns True on success.

Index

Name or index, starting at 0, of the category to remove

ries.Remove method

^{on}
^{cegory} from the library.
^{con}
^{cegory} from the library.
^{cegory} Variant) As Boolean

cess.

tarting at 0, of the category to remove

QueryCategory.Add method

<u>QueryCategory collection</u> <u>See Also</u> Saves the current query to the category or an external file.

Add(QueryName As String, _ QueryDescription As String, _ SaveOptions As Long _ QueryFileName As String _) As SavedQuery

Returns the saved query.

QueryName

Query name

QueryDescription

Query description

SaveOptions

Save options, one or more ESaveQueryOptions constants. This parameter is relevant only if there is a QueryFileName argument—that is, if the query is saved externally.

QueryFileName

Path and file name or the empty string. If a path and file name are given, the query is saved externally.

You combine ESaveQueryOptions constants with the + operator. For example, to save the coordinates of the query and the alter properties, the argument is kQrySaveCoordinates + kQrySaveAlteration.

ry.Add method

<u>See Also</u> ery to the category or an external file.

```
As String, __
As String, __
Long __
As String __
```

lery.

n

e or more ESaveQueryOptions constants. This parameter is here is a QueryFileName argument—that is, if the query is

e or the empty string. If a path and file name are given, the ternally.

QueryOptions constants with the + operator. For example, to of the query and the alter properties, the argument is s + kQrySaveAlteration.

QueryCategory.AddFromFile method

<u>QueryCategory collection</u> <u>See Also</u> Adds an external query to the category.

AddFromFile(_ QueryName As String, _ QueryDescription As String, _ QueryFileName As String _) As SavedQuery

Returns the saved query.

QueryName

Query name

QueryDescription

Query description

QueryFileName

Path and file name of the query to add

ry.AddFromFile method

ery to the category.

tring, _ 1 As String, _ As String _

lery.

n

ie of the query to add

QueryCategory.Item method

<u>QueryCategory collection</u> Gets a saved query from the category.

Item(Index As Variant) As SavedQuery

Returns the saved query.

QueryNameOrIndex

The name or index, starting at 0, of the query to get.

ry.Item method

¹ rom the category.

riant) As SavedQuery

lery.

ex

ex, starting at 0, of the query to get.

QueryCategory.Remove method

<u>QueryCategory collection</u> Removes a saved query from the category.

Remove(Index As Variant) As Boolean

Returns True on success.

Index

Name or index, starting at 0, of the query to remove.

ry.Remove method

^a ery from the category. **; Variant) As Boolean** cess.

tarting at 0, of the query to remove.

QueryLeaf.SetDataCond method

<u>QueryLeaf object</u> <u>Example</u> <u>See Also</u> Defines a data condition.

SetDataCond(_ DataType As EDataQueryType, _ CondOperation As EConditionOperator, _ Table As String, _ Field As String, _ Value As Variant _) As Boolean

Returns True on success.

DataType

The EDataQueryType constant that specifies the kind of data to query.

CondOperation

The EConditionOperator constant that specifies the comparison operator for the query. **Note** The only valid operator in a string context is kCondEq.

Table

A data source: block name, link template, or RegApp.

Field

A field value for the data type:

Attribute tag of a block Column name of an object data table Object data field name Extended entity data field name

Value

A value in the data source to query.

Note Call this method only if the QueryLeaf object's Type property is kDataCondition.

The following example performs a query for data less than the value in a table Water_Bodies, in a field called Avg_Depth with a value of 10.

Dim amap As AcadMap Dim prj As Project Dim qry As Query

```
Dim mainqrybr As QueryBranch
Dim andqrybr As QueryBranch
Dim dataqrylf As QueryLeaf
Dim qrylf As QueryLeaf
Dim boolVal As Boolean
Dim atdr As AttachedDrawing
Set amap = ThisDrawing.Application. _
   GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
Set qry = prj.CurrQuery
Set atdr = prj.DrawingSet.Add("MAPTUT:\\citymap7.dwg")
boolVal = qry.Clear
Set mainqrybr = qry.QueryBranch
Set qrylf = mainqrybr.Add( _
   kLocationCondition, _
   kOperatorAnd)
Set andqrybr = mainqrybr.Add( _
   kQueryBranch, _
   kOperatorAnd)
Set dataqrylf = andqrybr.Add( _
   kDataCondition, _
   kOperatorOr)
boolVal = qrylf.SetLocationCond( _
   kLocationInside, _
   prj.MapUtil.NewLocationAll)
boolVal = datagrylf.SetDataCond( _
   kDataIRD, _
   kCondLT, _
   "Water_Bodies", _
   "Avg_Depth", _
   10)
```
QueryLeaf.SetLocationCond method

<u>QueryLeaf object</u> <u>Example</u> <u>See Also</u> Defines a location condition.

SetLocationCond(_ DataType As ELocationType, _ BoundaryObj As Variant _) As Boolean

Returns True on success.

DataType

One of the following ELocationType constants.

kLocationInside kLocationCrossing

BoundaryObj

A boundary object, such as a PolygonBound.

Boundary objects are created by methods of the MapUtil object.

Note Call this method only if the QueryLeaf object's Type property is kLocationCondition.

etLocationCond method

<u>nple See Also</u> ondition.

cess.

ving ELocationType constants. le sing

ct, such as a PolygonBound.

e created by methods of the MapUtil object.

d only if the QueryLeaf object's Type property is

QueryLeaf.SetPropertyCond method

<u>QueryLeaf object</u> <u>Example</u> <u>See Also</u> Defines a property condition.

SetPropertyCond(_ Property As EPropertyType, _ CondOperation As EConditionOperator, _ Value As Variant _) As Boolean

Returns True on success.

Property

Specifies an entity property.

CondOperation

Specifies the comparison operator. **Note** The only valid operator in a string context is kCondEq.

Value

Specifies a string, long, or integer to use in a property condition.

Note Call this method only if the QueryLeaf object's Type property is kPropertyCondition.

The following example defines a property condition where the property is *layer*, the comparison operator is *equal to*, and the value is *Stream*.

Dim amap As AcadMap Dim prj As Project Dim qry As Query Dim mainqrybr As QueryBranch Dim andqrybr As QueryBranch Dim propqrylf As QueryLeaf Dim qrylf As QueryLeaf Dim boolVal As Boolean Dim atdr As AttachedDrawing

```
Set amap = ThisDrawing.Application. _
GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
Set qry = prj.CurrQuery
```

Set atdr = prj.DrawingSet.Add("MAPTUT:\\citymap7.dwg")

boolVal = qry.Clear

Set mainqrybr = qry.QueryBranch

Set qrylf = mainqrybr.Add(kLocationCondition, kOperatorAnd)

Set andqrybr = mainqrybr.Add(kQueryBranch, kOperatorAnd)

Set propqrylf = andqrybr.Add(kPropertyCondition, kOperatorOr)

boolVal = qrylf.SetLocationCond(kLocationInside, _

prj.MapUtil.NewLocationAll)

boolVal = propqrylf.SetPropertyCond(kLayer, kCondEq, "Stream")

QueryLeaf.SetSQLCond method

<u>QueryLeaf object</u> Defines an SQL condition.

SetSQLCond(_ LTNName As String, _ SQLWhere As String _) As Boolean

Returns True on success.

LTNName

Link Template name.

SQLWhere

SQL Where clause. (Don't include the Where keyword.)

Note Call this method only if the QueryLeaf object's Type property is kSQLCondition.

The following example defines an SQL condition.

Dim amap As AcadMap Dim prj As Project Dim qry As Query Dim atdr As AttachedDrawing Dim mainqrybr As QueryBranch Dim andqrybr As QueryBranch Dim sqlqrylf As QueryLeaf Dim qrylf As QueryLeaf Dim boolVal As Boolean Dim whereVal as String Dim strLT as String

```
Set amap = ThisDrawing.Application. _
GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
Set qry = prj.CurrQuery
Set atdr = prj.DrawingSet.Add("MAPTUT:\\citymap7.dwg")
boolVal = qry.Clear
Set mainqrybr = qry.QueryBranch
Set qrylf = mainqrybr.Add(kLocationCondition, kOperatorAnd)
```

```
Set andqrybr = mainqrybr.Add(kQueryBranch, kOperatorAnd)
Set sqlqrylf = andqrybr.Add(kDataCondition, kOperatorOr)
boolVal = qrylf.SetLocationCond(kLocationInside, _
prj.MapUtil.NewLocationAll)
strLT = "Link template name goes here"
whereVal = "Top = 500 AND Left = 250"
boolVal = sqlqrylf.SetSQLCond("LTname", WhereVal)
```

QueryReport.Add method

Adds an expression to the report template.

Add(Expression As String) As Expression

Returns an expression.

Expression

A string expression, evaluated when a query executes.

The following example adds two expressions to a template that produce a report listing the length and layer of each queried object.

```
Dim amap As AcadMap
Dim prj As Project
Dim atdr As AttachedDrawing
Dim qry As Query
Dim qrybr As QueryBranch
Dim qrylf As QueryLeaf
Dim qryrp As QueryReport
Dim exp As Expression
Dim exp2 As Expression
```

```
Set amap = ThisDrawing.Application.
GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
Set atdr = prj.DrawingSet.Add("MAPTUT:\\citymap7.dwg")
Set qry = prj.CurrQuery
gry.Clear
Set qrybr = qry.QueryBranch
Set grylf = grybr.Add(kLocationCondition, kOperatorAnd)
qrylf.SetLocationCond kLocationInside, prj.MapUtil.NewLocationAll
Set gryrp = prj.CurrQuery.Report
Set exp = qryrp.Add(".LENGTH")
Set exp2 = qryrp.Add(".LAYER")
prj.CurrQuery.Report.ReportFileName = "MyReport.txt"
qry.Mode = kQueryReport
qry.Define qrybr
gry.Execute
```

QueryReport.Clear method

<u>QueryReport collection</u> Removes all expressions from the template.

Clear As Boolean

Returns True on success.

The following example removes all expressions from a template.

Dim amap As AcadMap Dim prj As Project Dim atdr As AttachedDrawing Dim qry As Query Dim qrybr As QueryBranch Dim qrylf As QueryLeaf Dim qryrp As QueryReport

```
Set amap = ThisDrawing.Application. _
GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
Set atdr = prj.DrawingSet.Add("MAPTUT:\\citymap7.dwg")
Set qry = prj.CurrQuery
qry.Clear
Set qrybr = qry.QueryBranch
Set qrylf = qrybr.Add(kLocationCondition, kOperatorAnd)
qrylf.SetLocationCond kLocationInside, prj.MapUtil.NewLocationAll
Set qryrp = prj.CurrQuery.Report
qryrp.Clear
```

Clear method

ions from the template.

cess.

ple removes all expressions from a template.

∕Лар

dDrawing

Branch Leaf Report

wing.Application. _
'AutoCADMap.Application'')
cts(ThisDrawing)
ugSet.Add("MAPTUT:\\citymap7.dwg")
ery

yBranch d(kLocationCondition, kOperatorAnd) nd kLocationInside, prj.MapUtil.NewLocationAll Query.Report

QueryReport.Item method

<u>QueryReport collection</u> <u>See Also</u> Gets a specific expression from the report template.

Item(ExpNumber As Long) As Expression

Returns the line.

ExpNumber

The index number, or position, of the expression to get, starting at 0.

The following example prints the values of an expressions in a template.

```
Dim amap As AcadMap
Dim prj As Project
Dim qry As Query
Dim qrybr As QueryBranch
Dim qrylf As QueryLeaf
Dim qryrp As QueryReport
Dim exp As Expression
Dim exp2 As Expression
Dim i As Integer
```

```
Set amap = ThisDrawing.Application. _
GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
Set qry = prj.CurrQuery
qry.Clear
Set qrybr = qry.QueryBranch
Set qrylf = qrybr.Add(kLocationCondition, kOperatorAnd)
qrylf.SetLocationCond kLocationInside, prj.MapUtil.NewLocationAll
Set qryrp = prj.CurrQuery.Report
qryrp.Clear
Set exp = qryrp.Add(".LENGTH")
Set exp2 = qryrp.Add(".LAYER")
For i = 0 To qryrp.Count - 1
Debug.Print qryrp.Item(i).Value
Next i
```

QueryReport.Remove method

<u>QueryReport collection</u> Removes an expression from a report template.

Remove(ExpNumber As Long) As Boolean

Returns True on success.

ExpNumber

The index number, or position, of the expression to remove, starting at 0.

The following example removes the .LAYER expression from the report template.

Dim amap As AcadMap Dim prj As Project Dim atdr As AttachedDrawing Dim qry As Query Dim qrybr As QueryBranch Dim grylf As QueryLeaf Dim gryrp As QueryReport Dim exp As Expression Dim exp2 As Expression Dim i As Integer Set amap = ThisDrawing.Application. GetInterfaceObject("AutoCADMap.Application") Set prj = amap.Projects(ThisDrawing) Set atdr = prj.DrawingSet.Add("MAPTUT:\\citymap7.dwg") Set qry = prj.CurrQuery qry.Clear Set qrybr = qry.QueryBranch Set qrylf = qrybr.Add(kLocationCondition, kOperatorAnd) grylf.SetLocationCond kLocationInside, prj.MapUtil.NewLocationAll Set gryrp = prj.CurrQuery.Report Set exp = qryrp.Add(".LENGTH") Set exp2 = qryrp.Add(".LAYER") prj.CurrQuery.Report.ReportFileName = "MyReport.txt" qry.Mode = kQueryReport For i = 0 To gryp.Count - 1 If gryrp.Item(i).Value = ".LAYER" Then

qryrp.Remove (i) Exit For End If Next i qry.Define qrybr qry.Execute

RangeTable.Add method

Adds a range to the range table.

Add(ExprValue As String, _ ReturnValue As String, _ CompareOperation As ERangeOperator _) As Range

Returns a range.

ExprValue

The comparison value of the range's condition.

ReturnValue

The value to return if the range's condition is true.

CompareOperation

The comparitive operator of the range's condition.

This example is an excerpt from the sample code for altering queried objects. To go to the sample code, click . The following example shows how to set up a range table called safewater. The query uses the range table to specify a range of less than 10 feet and to return the name of the STREAM layer. The last line of code adds a property alteration to the AlterLines collection that specifies what to do with the range and return value--to move water bodies having a depth within that range to the STREAM layer.

Dim rngtb As RangeTable Dim rng As Range Dim altlLay As AlterLine

'Alter Layer Based on Range Table
prj.RangeTables.Remove ("foulwater")
Set rngtb = prj.RangeTables.Add("safewater", "shallow water")
Set rng = rngtb.Add("10", "STREAM", kRangeLT)
Set altlLay = prj.CurrQuery.AlterProp.Add(_
kAlterationLayer, _
"(Range :AVG_DEPTH@WATER_BODIES foulwater)")

RangeTable.Item method

RangeTable collection Gets a range from the range table.

Item(Index As Long) As Range

Returns the range.

Index

The index, starting at 0, of a range in the range table.

tem method

e range table. ng) As Range

ıg at 0, of a range in the range table.

RangeTable.Remove method

Range Table collection Removes a range from the range table.

Remove(Index As Long) As Boolean

Returns True on success.

Index

The index, starting at 0, of a range in the range table.

The following example is a procedure that you call from the sample code for altering queried objects. To go to the sample code, click **D**. The procedure traverses the collection of RangeTables looking for a table that contains the range that specifies the "MARSH" return value, and removes it. Add a call to rmrng just before the call to qry.Define in the sample code.

rmrng prj, rngtb

```
'Define Query
boolVal = qry.Define(mainqrybr)
.
.
.
.
Sub rmrng(prj As Project, rngtb As RangeTable)
Dim i As Integer
Dim rng As Range
For i = 0 To prj.RangeTables.Count - 1
For Each rng In prj.RangeTables.Item(i)
If rng.ReturnValue = "MARSH" Then
prj.RangeTables.Remove (i)
End If
Next
```

Next i

End Sub

Remove method

m the range table.

; Long) As Boolean

cess.

ng at 0, of a range in the range table.

ple is a procedure that you call from the sample code for cts. To go to the sample code, click **>**. The procedure on of RangeTables looking for a table that contains the he "MARSH" return value, and removes it. Add a call to call to qry.Define in the sample code.

e(mainqrybr)

oject, rngtb As RangeTable)

geTables.Count - 1 RangeTables.Item(i) "MARSH" Then nove (i)

RangeTables.Add method

<u>RangeTables collection</u> <u>Example</u> Adds a range table to the collection.

Add(RangeTblName As String, _ RangeTblDescription As String _) As RangeTable

Returns the added range table.

RangeTbleName

Name of the range table.

RangeTblDescription

Description of the range table.

The following excerpt from the sample code for altering queried objects removes a table of the same name as the one you want to add, if it exists, and adds a range table called foulwater to the range tables collection. To see the complete example, click **N**.

Dim prj As Project Dim rngtb As RangeTable

```
prj.RangeTables.Remove ("foulwater")
Set rngtb = prj.RangeTables.Add("foulwater", "shallow water")
```

Add method

) the collection.

nge table.

;e table.

tion

e range table.

pt from the sample code for altering queried objects removes ame as the one you want to add, if it exists, and adds a range r to the range tables collection. To see the complete

Table

```
nove ("foulwater")
eTables.Add("foulwater", "shallow water")
```

RangeTables.Item method

RangeTables collection Gets a range table from the collection.

Item(Index As Variant) As RangeTable

Returns the range table.

Index

The name or index, starting at 0, of a range table in a range tables collection.

Item method

om the collection. riant) As RangeTable ole.

ex, starting at 0, of a range table in a range tables collection.

RangeTables.Remove method

RangeTables collection Removes a range table from the collection.

Remove(Index As Variant) As Boolean

Returns True on success.

Index

The name or index, starting at 0, of a range table in a range tables collection.
Remove method

le from the collection.

; Variant) As Boolean

cess.

ex, starting at 0, of a range table in a range tables collection.

SaveSet.AddObjects method

SaveSet collection Adds objects to the save set.

AddObjects(ToBeAdded As Variant) As Long

Returns the number of objects added.

ToBeAdded

An array of Longs containing the IDs of objects to be added to a save set.

You can add the IDs of locked, erased, or newly created objects.

The following example assumes you altered queried objects of a drawing before running this code. For sample code for querying a drawing and altering properties, For more information, click **N**. This example adds queried objects to a save set and displays their IDs.

Dim amap As AcadMap Dim prj As Project Dim i As Integer Dim entry As Object Dim varArray As Variant Dim lngIDArray() As Long ReDim lngIDArray(ThisDrawing.ModelSpace.Count) Dim strOutput As String

```
Set amap = ThisDrawing.Application. _
GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
For Each entry In ThisDrawing.ModelSpace
IngIDArray(i) = entry.ObjectID
i = i + 1
Next
prj.SaveSet.AddObjects IngIDArray
varArray = prj.SaveSet.GetObjects(kQueriedExisted)
Debug.Print "SaveSet objects--all queried objects"
If UBound(varArray) > -1 Then
For i = 0 To UBound(varArray)
Debug.Print varArray(i)
Next i
End If
```

SaveSet.GetObjects method

 $\begin{matrix} \frac{SaveSet \ collection}{Gets \ the \ IDs \ of \ objects \ in \ the \ save \ set. \end{matrix}$

GetObjects(Qualifier As ESaveSetObjectType) As Variant

Returns the IDs in an array of Long.

Qualifier

The types of save set objects you want to get, one or more ESaveSetObjectType constants.

You can combine constants to specify more than one type. For example, to return the queried objects that have not been deleted and the new objects, the Qualifier argument is kQueriedExisted + kNewlyCreated.

The following example assumes you altered queried objects of a drawing before running this code. For sample code for querying a drawing and altering properties, click . This example adds queried objects to a save set and displays their IDs.

Dim amap As AcadMap Dim prj As Project Dim i As Integer Dim entry As Object Dim varArray As Variant Dim lngIDArray() As Long ReDim lngIDArray(ThisDrawing.ModelSpace.Count) Dim strOutput As String

```
Set amap = ThisDrawing.Application. _
GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
For Each entry In ThisDrawing.ModelSpace
IngIDArray(i) = entry.ObjectID
i = i + 1
Next
prj.SaveSet.AddObjects IngIDArray
varArray = prj.SaveSet.GetObjects(kQueriedExisted)
Debug.Print "SaveSet objects--all queried objects"
If UBound(varArray) > -1 Then
For i = 0 To UBound(varArray)
```

Debug.Print varArray(i) Next i End If

SaveSet.RemoveObjects method

SaveSet collection Removes an object from a save set.

RemoveObjects(ToBeRemoved As Variant) As Long

Returns the number of objects removed from the save set.

ToBeRemoved

An array of Longs containing the IDs of objects in the save set.

You can specify the IDs of locked, erased, or newly created objects.

The following example builds on the SaveSet.Add example. For more information, click ■.

You call the following procedure from the SaveSet.Add example to remove three objects from the save set.

Sub removeobjects(lngIDArray As Variant)

Dim amap As AcadMap Dim prj As Project Dim i As Integer Dim varArray As Variant Dim strOutput As String Dim lngIDArray2() As Long ReDim lngIDArray2(3)

```
Set amap = ThisDrawing.Application. _
GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
IngIDArray2(0) = IngIDArray(1)
IngIDArray2(1) = IngIDArray(5)
IngIDArray2(3) = IngIDArray(8)
prj.SaveSet.removeobjects IngIDArray2
varArray = prj.SaveSet.GetObjects(kQueriedExisted)
Debug.Print "Three objects removed from save set"
If UBound(varArray) > -1 Then
For i = 0 To UBound(varArray)
Debug.Print varArray(i)
Next i
End If
```

End Sub

AcadMap.Aliases property

Read Only See Also

Aliases As Aliases

Returns the drive aliases collection.

Using this object, you get a collection of shorthand names of drives and directory locations.

lases property

lso

S

ases collection.

u get a collection of shorthand names of drives and

AcadMap.ErrorStack property

Read Only Example See Also

ErrorStack As ErrorStack

Returns the error stack collection.

orStack property

<u>ple</u> <u>See Also</u>

orStack

ck collection.

AcadMap.Projects property

Read Only See Also

Projects As Projects

Returns the projects collection.

Using this collection, you get a Project, and access another level of AutoCAD Map functionality through the query, drawing set, object data tables, project options, and other objects.

jects property

lso

<u>:ts</u>

collection.

, you get a Project, and access another level of AutoCAD rough the query, drawing set, object data tables, project ojects.

AcadMap.SystemOptions property

Read Only See Also

SystemOptions As SystemOptions

Returns the system options.

stemOptions property

lso

s SystemOptions

ptions.

Alias.Directory property

<u>Alias object</u> Read-Write

Directory As String

Sets or returns the aliased path.

ry property

١g

iased path.

Alias.Name property

<u>Alias object</u> Read-Write

Name As String

Sets or returns the alias name.

roperty

ias name.

Aliases.Count property

Read Only See Also

Count As Long

Returns the number of aliases in an AcadMap object.

Use this property to list current drive aliases.

The following example displays the name of each alias.

Dim als As Alias Dim cAls As Integer, i As Integer Dim amap As AcadMap Dim strOutput As String

```
Set amap = ThisDrawing.Application. _
GetInterfaceObject("AutoCADMap.Application")
cAls = amap.Aliases.Count
For i = 0 To cAls - 1
Set als = amap.Aliases.Item(i)
strOutput = strOutput & als.Name & "=" & als.Directory & Chr(13)
Next i
MsgBox strOutput
```

t property

<u>Also</u>

of aliases in an AcadMap object. list current drive aliases. ple displays the name of each alias.

', i As Integer ∕Iap ring

```
ving.Application. _
'AutoCADMap.Application'')
.Count
```

```
es.Item(i)
It & als.Name & "=" & als.Directory & Chr(13)
```

AlterLine.Value property

Read-Write See Also

Value As Variant

Sets or returns the new value for the property, or an expression that evaluates to such a value.

The value's type must be appropriate for the property. For example, if AlterLine.Property is kAlterationElevation, the value should be a point. If AlterLine.Property is kAlterationHeight, kAlterationRotation, kAlterationWidth, or kAlterationThickness, the value should be a real. For all other kinds of properties, the value should be a string that makes sense with such a property.

If you are using a range table to provide values conditionally, use AlterLine.Value to reference the range table by supplying a range expression as argument. For more information, click \blacksquare .

```
The following procedure traverses a collection of property alterations and prints
the value of the alteration if it is a text definition. Call this procedure just before
the call to gry. Define, as shown in the following example. To go to the sample
code, click .
'Call from Altering Queried Objects sample
showtxtdf prj, altls
'Define Query
boolVal = gry.Define(maingrybr)
Sub showtxtdf(prj As Project, altls As AlterLines)
Dim i As Integer
Dim altl As Variant
Dim txtdf As TextDef
For i = 0 To prj.CurrQuery.AlterProp.Count - 1
Set altl = altls.Item(i)
If altl.Property = kAlterationTextEntity Then
Set txtdf = altls.Item(i)
Debug.Print txtdf.TextValue
```

End If

Next i End Sub

AlterLines.Count property

AlterLines collection Read Only

Count As Long

Returns the number of property alterations in the collection.

ount property

of property alterations in the collection.
AttachedDrawing.Active property

AttachedDrawing object See Also Read-Write

Active As Boolean

Sets or returns True if an attached drawing is active for querying.

To get information on nested drawings or to get a drawing object from a nested drawing, you must activate the parent drawing.

wing.Active property

See Also

n

if an attached drawing is active for querying.

n nested drawings or to get a drawing object from a nested ctivate the parent drawing.

AttachedDrawing.AliasPath property

AttachedDrawing object Read Only

AliasPath As String

Returns the drive alias that represents a hard-coded drive name and path.

wing.AliasPath property

١g

as that represents a hard-coded drive name and path.

AttachedDrawing.ApplyTransform property

AttachedDrawing object See Also Read-Write

ApplyTransform As Boolean

Sets or returns True when queried objects are transformed.

wing.ApplyTransform property

See Also

As Boolean

when queried objects are transformed.

AttachedDrawing.Description property

AttachedDrawing object Read-Write

Description As String

Sets or returns a description of the attached drawing.

wing.Description property

ring

cription of the attached drawing.

AttachedDrawing.Extents property

Read Only See Also

Extents As Variant

Returns the attached drawing's extents as an array of four Doubles.

Coordinates are returned in the following order: MinX, MinY, MaxX, MaxY. The extents of the attached drawing is the smallest rectangle that encompasses all the objects in it. (MinX, MinY) is the lower-left corner, and (MaxX, MaxY) is the upper right.

The attached drawing must be active when you use this property.

The following code is an excerpt from a larger example. The code shows how to create the main branch of a query of all objects within the extents of the attached drawing. For the entire example, click \blacksquare .

```
Dim amap As AcadMap
Dim prj As Project
Dim qry As Query
Dim mainqrybr As QueryBranch
Dim qrylf As QueryLeaf
Dim mapu As MapUtil
Dim wind As WindowBound
Dim boolVal As Boolean
Dim dblary As Variant
```

```
Set amap = ThisDrawing.Application. _
GetInterfaceObject("AutoCADMap.Application")
Set prj = amap.Projects(ThisDrawing)
Set qry = prj.CurrQuery
```

AttachedDrawing.HasLocks property

AttachedDrawing objectSee AlsoRead Only

HasLocks As Boolean

Returns True if an attached drawing contains locked objects.

wing.HasLocks property

See Also

lean

tached drawing contains locked objects.

AttachedDrawing.HasNested property

AttachedDrawing object See Also Read Only

HasNested As Boolean

Returns True if the drawing is indirectly attached to the project.

wing.HasNested property

See Also

olean

rawing is indirectly attached to the project.

AttachedDrawing.IsTopLevel property

AttachedDrawing object Read Only

IsTopLevel As Boolean

Returns True if this drawing is directly attached to the project.

wing.IsTopLevel property

olean

lrawing is directly attached to the project.

AttachedDrawing.NestedDrawings property

AttachedDrawing object See Also Read Only

NestedDrawings As NestedDrawings

Returns the attached drawing's NestedDrawings collection.

wing.NestedDrawings property

<u>See Also</u>

As NestedDrawings

drawing's NestedDrawings collection.

AttachedDrawing.Parent property

Read Only See Also

Parent As AttachedDrawing

Returns a reference to the parent of the attached drawing.

Used only if the attached drawing is part of the NestedDrawings collection of another attached drawing, which is referred to as its *parent* drawing.

*v*ing.Parent property

See Also

dDrawing

to the parent of the attached drawing.

ched drawing is part of the NestedDrawings collection of wing, which is referred to as its *parent* drawing.

AttachedDrawing.Projection property

AttachedDrawing object See Also Read-Write

Projection As String

Sets or returns the name of the drawing coordinate system.

You can change the coordinate system of an attached drawing only if no objects in the drawing have been queried in the current session *and* (in a multi-user environment) if no other user embeds locked objects in the drawing.

wing.Projection property

See Also

ng

me of the drawing coordinate system.

coordinate system of an attached drawing only if no objects been queried in the current session *and* (in a multi-user ther user embeds locked objects in the drawing.

AttachedDrawing.SaveBackExt property

AttachedDrawing object See Also Read-Write

SaveBackExt As Variant

Sets or returns the attached drawing's save-back extents as an array of four Doubles.

Coordinates are returned in the following order: MinX, MinY, MaxX, MaxY. Objects in the save set that are associated with the attached drawing are saved back only if they are within or crossing the save-back extents boundary. (MinX, MinY) is the lower-left corner, and (MaxX, MaxY) is the upper right.

The attached drawing must be active when you use this property.

wing.SaveBackExt property

See Also

Variant

tached drawing's save-back extents as an array of four

rned in the following order: MinX, MinY, MaxX, MaxY. et that are associated with the attached drawing are saved within or crossing the save-back extents boundary. (MinX, eft corner, and (MaxX, MaxY) is the upper right.

g must be active when you use this property.
AttachedDrawing.Trotate property

AttachedDrawing object See Also Read-Write

Trotate As Double

Sets or returns the simple transform rotation of the attached drawing.

Returns True if rotation is clockwise.

wing.Trotate property

<u>See Also</u>

j

mple transform rotation of the attached drawing. ion is clockwise.

AttachedDrawing.Tscale property

AttachedDrawing object See Also Read-Write

Tscale As Double

Sets or returns the simple transform scale of an attached drawing.

The transformation temporarily scales the queried objects.

wing.Tscale property

See Also

mple transform scale of an attached drawing. emporarily scales the queried objects.

AttachedDrawing.TXOffset property

AttachedDrawing object See Also Read-Write

TXOffset As Double

Sets or returns the X offset portion of the X and Y transformation coordinates for the drawing.

wing.TXOffset property

See Also

ble

offset portion of the X and Y transformation coordinates for

AttachedDrawing.TYOffset property

AttachedDrawing object See Also Read-Write

TYOffset As Double

Sets of returns the Y offset portion of the X and Y transformation coordinates for the drawing.

wing.TYOffset property

See Also

ble

offset portion of the X and Y transformation coordinates for

BufferFenceBound.Count property

BufferFenceBound collection Read Only

Count As Long

Returns the count of the total number of points, or Point3d objects in a BufferFenceBound collection.

3ound.Count property

<u>ction</u>

the total number of points, or Point3d objects in a collection.

BufferFenceBound.Width property

BufferFenceBound collection Read-Write

Width As Double

Sets or returns the width of the buffer zone.

3ound.Width property

<u>ction</u>

idth of the buffer zone.

BufferPolylineBound.Count property

BufferPolylineBound collection Read Only

Count As Long

Returns the number of points in the BufferPolylineBound collection.

eBound.Count property

<u>lection</u>

of points in the BufferPolylineBound collection.

BufferPolylineBound.ObjectID property

BufferPolylineBound collection Read-Write

ObjectID As Long

Sets or returns the ID the BufferPolylineBound collection.

eBound.ObjectID property

<u>lection</u>

3

) the BufferPolylineBound collection.

BufferPolylineBound.Width property

BufferPolylineBound collection Read-Write

Width As Double

Sets or returns the width of the buffer zone.

eBound.Width property

<u>lection</u>

idth of the buffer zone.

CircleBound.CenterX property

<u>CircleBound object</u> Read-Write

CenterX As Double

Sets or returns the center X coordinate of a CircleBound object.

The following example prints the center of a CircleBound object in the Immediate Window.

Dim acadapp As AcadApplication Dim amap As AcadMap Dim prj As Project Dim cir As CircleBound Dim acadu As AcadUtility Dim varVal As Variant, Radius As Double, boolVal As Boolean Dim strVal As String, ReturnValue As String

Set acadapp = ThisDrawing.Application Set amap = acadapp.GetInterfaceObject("AutoCADMap.Application") Set prj = amap.Projects(ThisDrawing) Set acadu = acadapp.ActiveDocument.Utility varVal = acadu.GetPoint(, "Select centre of circle: ") Radius = acadu.GetDistance(varVal, "Drag a line for radius: ") Set cir = prj.MapUtil.NewCircle(Radius, varVal(0), varVal(1)) Debug.Print cir.CenterX

CenterX property

le

enter X coordinate of a CircleBound object. ple prints the center of a CircleBound object in the

adApplication

und Utility Int, Radius As Double, boolVal As Boolean ;, ReturnValue As String

'rawing.Application GetInterfaceObject("AutoCADMap.Application") cts(ThisDrawing) .ActiveDocument.Utility oint(, "Select centre of circle: ") Distance(varVal, "Drag a line for radius: ") l.NewCircle(Radius, varVal(0), varVal(1)) erX

CircleBound.CenterY property

<u>CircleBound object</u> Read-Write

CenterY As Double

Sets or returns the center Y coordinate of a CircleBound object.

CenterY property

le

nter Y coordinate of a CircleBound object.

CircleBound.CenterZ property

<u>CircleBound object</u> Read-Write

CenterZ As Double

Sets or returns the center Z coordinate of a Circlebound object (ignored).

CenterZ property

le

enter Z coordinate of a Circlebound object (ignored).
CircleBound.Radius property

<u>CircleBound object</u> Read-Write

Radius As Double

Sets or returns the radius of a CircleBound object.

Radius property

) ,

dius of a CircleBound object.

ClosedPolylineBound.Count property

<u>ClosedPolylineBound collection</u> Read Only

Count As Long

Sets or returns number of points in the ClosedPolylineBound collection.

neBound.Count property

<u>llection</u>

er of points in the ClosedPolylineBound collection.

ClosedPolylineBound.ObjectID property

<u>ClosedPolylineBound collection</u> Read-Write

ObjectID As Long

Sets or returns the ID of the ClosedPolylineBound collection.

neBound.ObjectID property

<u>llection</u>

3

) of the ClosedPolylineBound collection.

DiagParam.Message property

Read Only See Also

Message As String

Returns a description of the error.

lessage property

<u>Also</u>

g

ι of the error.

DiagParam.Position property

Read Only See Also

Position As Long

Returns the position of the error within the SQL statement.

osition property

<u>Also</u>

of the error within the SQL statement.

DiagParam.Source property

Read Only See Also

Source As EAdeClassId

Returns the identifier of the ADE object that caused the error.

ource property

<u>Also</u>

lassId

r of the ADE object that caused the error.

DiagParam.SqlStatement property

Read Only See Also

SqlStatement As String

Returns the SQL statement that caused the error.

An empty string means SQL processing did not cause the error.

qlStatement property

<u>Also</u>

String

ement that caused the error.

INS SQL processing did not cause the error.

DrawingSet.Count property

Read Only See Also

Count As Long

Returns the number of drawings in the drawing set.

Note The Count property does not include nested drawings.

Count property

<u>See Also</u>

of drawings in the drawing set. Derty does not include nested drawings.

DrawingSet.Extents property

Read Only See Also

Extents As Variant

Returns the extents of all active drawings.

Returns coordinates as an array of four Doubles in the following order: MinX, MinY, MaxX, MaxY. The extents of the drawing set is the smallest rectangle that encompasses all the objects in all the attached and active drawings. (MinX, MinY) is the lower-left corner, and (MaxX, MaxY) is the upper right.

Extents property

See Also

t

of all active drawings.

as an array of four Doubles in the following order: MinX, The extents of the drawing set is the smallest rectangle that objects in all the attached and active drawings. (MinX, eft corner, and (MaxX, MaxY) is the upper right.

ErrorEntry.Count property

Read Only See Also

Count As Long

Returns the number of diagnostic parameters associated with the error.

ount property

ee Also

of diagnostic parameters associated with the error.

ErrorEntry.ErrCode property

Read Only See Also

ErrCode As EErrCode

Returns the error number.

For a list of error codes alphabetically by name, see the object browser in the VBA editor. Each name entry also shows the corresponding number.

For a list of error codes by number, click **Z**.

rCode property

ee Also

Code

nber.

les alphabetically by name, see the object browser in the me entry also shows the corresponding number.

les by number, click 🗷.

ErrorEntry.ErrMessage property

Read Only See Also

ErrMessage As String

Returns a description of the error.

rMessage property

<u>ee Also</u>

ring

ι of the error.
ErrorEntry.ErrType property

Read Only See Also

ErrType As EErrType

Returns the type of the error.

rType property

<u>ee Also</u>

ype

he error.

ErrorStack.Count property

ErrorStack collection Read Only

Count As Long

Returns the number of error messages in the error stack.

The count starts at 0.

ount property

,

of error messages in the error stack.

ErrorStack.LastErrCode property

Read Only See Also

LastErrCode As EErrCode

Returns the error number of the latest error that was pushed to the stack by AutoCAD Map.

Read this value immediately after the error occurs. This value is valid only after calling the method or using the property that caused the error.

For a list of error codes by number, click .

For a list of error codes alphabetically by name, see the object browser in the VBA editor. Each name entry in the object browser includes the corresponding number.

astErrCode property

ee Also

ErrCode

nber of the latest error that was pushed to the stack by

ediately after the error occurs. This value is valid only after r using the property that caused the error.

les by number, click 🗷.

les alphabetically by name, see the object browser in the me entry in the object browser includes the corresponding

Expression.Value property

Expression object Read-Write

Value As String

Sets or returns an expression, such as ".LENGTH" and ".COLOR".

alue property

pression, such as ".LENGTH" and ".COLOR".

FenceBound.Count property

<u>FenceBound collection</u> Read Only

Count As Long

Returns a count of the total number of points in a FenceBound collection.

Count property

le total number of points in a FenceBound collection.

HatchDef.Color property

HatchDef object Read-Write

Color As String

Sets or returns the name of the hatch color.

lor property

me of the hatch color.

HatchDef.Layer property

HatchDef object Read-Write

Layer As String

Sets or returns the name of the layer for the hatch.

yer property

me of the layer for the hatch.

HatchDef.Pattern property

Read-Write See Also

Pattern As String

Sets or returns a hatch pattern name or expression.

If you are using a range table to provide hatch patterns conditionally, use TextDef.TextValue to reference the range table by supplying a range expression as an argument. For more information, click **2**.

The following example consists of two procedures that you call from the sample code for altering queried objects. To go to the sample code, click . The first procedure sets a hatch pattern to ANSI36, and the second changes the pattern to ANSI38. Add a call to chghat1 just before the call to qry.Define in the sample code. Run the sample code and look at the Actrix drawing that displays the water bodies with the ANSI36 hatch pattern. Delete the call to chghat1 from the sample code, and add a call to chghat2. Rerun the sample code.

Sub chghat1(prj As Project, altls As AlterLines)

```
Dim i As Integer
Dim altl As Variant
Dim hatdf As HatchDef
Set hatdf = altls.AddHatchDef("ANSI36", _
"600", _
"90", _
"WATER", _
"CYAN")
End Sub
```

Sub chghat2(prj As Project, altls As AlterLines)

Dim i As Integer Dim altl As Variant Dim hatdf As HatchDef For i = 0 To prj.CurrQuery.AlterProp.Count - 1 Set altl = altls.Item(i) If altl.Property = kAlterationHatch Then Set hatdf = altls.Item(i) hatdf.Pattern = "ANSI38" End If Next i

End Sub

HatchDef.Property property

HatchDef object Read Only

Property As EAlterationType

The alteration type

For hatch alterations, there is only one alteration type: kAlterationHatch.

perty property

erationType

, there is only one alteration type: kAlterationHatch.

HatchDef.Rotation property

HatchDef object Read-Write

Rotation As String

Sets or returns the amount or expression to rotate the hatch.

tation property

3

nount or expression to rotate the hatch.

HatchDef.ScaleFactor property

HatchDef object Read-Write

ScaleFactor As String

Sets or returns the amount or expression to scale the hatch.

aleFactor property

ring

nount or expression to scale the hatch.

NestedDrawings.Count property

NestedDrawings collection Read Only

Count As Long

Returns the number of nested drawings in an attached drawing.

ngs.Count property

<u>on</u>

of nested drawings in an attached drawing.
ODFieldDef.Default property

ODFieldDef object Read-Write

Default As Variant

Sets or returns the default value of the field.

Default property

t

all value of the field.

ODFieldDef.Description property

ODFieldDef object Read-Write

Description As String

Sets or returns the description of the field.

Description property

ring

escription of the field.

ODFieldDef.IsUpdatable property

ODFieldDef object Read Only

IsUpdatable As Boolean

Returns True if you can update the field definition.

IsUpdatable property

oolean

can update the field definition.

ODFieldDef.Name property

ODFieldDef object Read-Write

Name As String

Sets or returns the name of the field.

Name property

me of the field.

ODFieldDef.Type property

Read Only See Also

Type As EDataType

Returns the data type of the field.

Type property

<u>Also</u>

pe

e of the field.

ODFieldDefs.Count property

ODFieldDefs collection Read Only

Count As Long

Returns the number of fields in the collection of fields.

.Count property

of fields in the collection of fields.

ODFieldDefs.IsUpdatable property

ODFieldDefs collection See Also Read Only

IsUpdatable As Boolean

Returns True if you can update the collection.

.IsUpdatable property

See Also

oolean

can update the collection.

ODFieldValue.Type property

Read Only See Also

Type As EDataType

Returns the field's data type.

e.Type property

ō

pe

ıta type.

ODFieldValue.Value property

<u>ODFieldValue</u> Read-Write

Value As Variant

Sets or returns the field's value.

e.Value property

eld's value.

ODRecord.Count property

ODRecord object Read Only

Count As Long

Returns the number of fields in the record.

ount property

of fields in the record.

ODRecord.ObjectID property

ODRecord object Read Only

ObjectID As Long

Returns the ID of the object to which this record is attached.

Returns 0 if the record is not attached to any object.

bjectID property

3

• object to which this record is attached.rd is not attached to any object.

ODRecord.TableName property

Read Only See Also

TableName As String

Returns the name of the table to which the record belongs.

ableName property

<u> Also</u>

ring

the table to which the record belongs.
ODRecords.IsDone property

ODRecords collection Read Only

IsDone As Boolean

Returns True if the current record is the last one.

Check this property as you traverse a list of records to prevent reading past the end of the list.

sDone property

ın

urrent record is the last one.

as you traverse a list of records to prevent reading past the

ODRecords.OwnerDbID property

ODRecords collection Read Only

OwnerDbID As Long

Returns the ID of the object to which the current record is attached.

The ID is specified when the object is created.

)wnerDbID property

ong

t to which the current record is attached.

e object is created.

ODTable.Description property

ODTable object Read Only

Description As String

Returns the description of the table.

cription property

ring

on of the table.

ODTable.Name property

ODTable object Read Only

Name As String

Returns the name of the table.

ne property

the table.

ODTable.ODFieldDefs property

Read Only See Also

ODFieldsDefs As ODFieldDefs

Returns the table's field definitions.

FieldDefs property

<u>so</u>

ODFieldDefs

eld definitions.

ODTable.StoreAsXdata property

ODTable object Read Only

StoreAsXdata As Boolean

Returns how record data is stored with drawing objects.

True: Record data is stored as extended entity data.

False: Record data is stored as AutoCAD Xrecords.

reAsXdata property

Boolean

data is stored with drawing objects. stored as extended entity data. stored as AutoCAD Xrecords.

ODTables.Count property

ODTables collection Read Only

Count As Long

Returns the number of tables in the project.

unt property

of tables in the project.

Point3d.X property

Point3d object Read-Write

X As Double

Sets or returns the X coordinate of a Point3d object.



coordinate of a Point3d object.

Point3d.Y property

Point3d object Read-Write

Y As Double

Sets or returns the Y coordinate of a Point3d object.



coordinate of a Point3d object.

Point3d.Z property

Point3d object Read-Write

Z As Double

Sets or returns the Z coordinate of a Point3d object (ignored).



coordinate of a Point3d object (ignored).

PolygonBound.Count property

PolygonBound collection Read Only.

Count As Long

Returns the count of the total number of points in a PolygonBound object.

Id.Count property

ļ

the total number of points in a PolygonBound object.

PolylineBound.Count property

PolylineBound collection Read Only

Count As Long

Returns the number of points in a PolylineBound collection.

Id.Count property

ļ

of points in a PolylineBound collection.
PolylineBound.ObjectID property

PolylineBound collection Read-Write

ObjectID As Long

Sets or returns the ID of the PolylineBound collection.

Id.ObjectID property

ļ

3) of the PolylineBound collection.

Project.CurrentProjection property

Read-Write See Also

CurrentProjection As String

Sets or returns the name of the project's coordinate system.

ntProjection property



1

ame of the project's coordinate system.

Project.CurrQuery property

Read Only See Also

CurrQuery As Query

Returns the current query.

Juery property

!

ery

luery.

Project.DrawingSet property

Read Only See Also

DrawingSet As DrawingSet

Returns the drawing set, which is a collection of attached drawings.

ingSet property

rawingSet

1

set, which is a collection of attached drawings.

Project.IsMapActiveCommand property

Project object Read Only

IsMapActiveCommand As Boolean

Returns True if an AutoCAD Map command is active.

oActiveCommand property

mand As Boolean

utoCAD Map command is active.

Project.MapUtil property

Read Only See Also

MapUtil As MapUtil

Returns the factory for query components.

Jtil property

Jtil

!

or query components.

Project.ODTables property

Read Only See Also

ODTables As ODTables

Returns the object data tables collection.

ibles property

Tables

1

ata tables collection.

Project.ProjectOptions property

Read Only See Also

ProjectOptions As ProjectOptions

Returns project options.

ctOptions property

1

; ProjectOptions

ms.

Project.QueryCategories property

Read Only See Also

QueryCategories As QueryCategories

Returns the Query Library, which is a collection of query categories.

7Categories property

As QueryCategories

!

ibrary, which is a collection of query categories.

Project.RangeTables property

Read Only See Also

RangeTables As RangeTables

Returns a collection of range tables.

eTables property

!

langeTables

of range tables.

Project.SaveSet property

Read Only See Also

SaveSet As SaveSet

Returns the save set collection.

You use this object to mark the objects to save back.

Set property

!

jet

collection.

c mark the objects to save back.

Projects.Count property

Read Only See Also

Count As Long

Returns the number of available AutoCAD Map projects.

ıt property

<u>Also</u>

of available AutoCAD Map projects.
Query.AlterDefined property

Read Only See Also

AlterDefined As Boolean

Returns True if Query.AlterProp defines a property alteration.

False if AlterProp is empty.

efined property

3oolean

y.AlterProp defines a property alteration. empty.

Query.AlterProp property

Read-Only Example See Also

AlterProp As AlterLines

Returns a property alteration definition.

To define property alteration, use AlterLines.Add, which assigns a valid AlterLines object to this property.To determine the status of property alteration, use Query.AlterDefined.

The following excerpt from a larger example uses the Query.AlterProp to clear property alterations that might have been created before running the example. For the complete example, click **B**.

Dim qry As Query Dim altls As AlterLines Dim altl As AlterLine Dim txtdf As TextDef ' Create a query -- code not shown Set altls = qry.AlterProp altls.RemoveAll

rop property

See Also

rLines

lteration definition.

lteration, use AlterLines.Add, which assigns a valid this property.To determine the status of property alteration, ned.

nes ie 2f ide not shown 'rop

Query.AlterTag property

Read-Write See Also

AlterTag As Boolean

Sets or returns whether the query alters retrieved objects.

To make alterations, you must set this flag to True before defining a query.

ag property

ean

er the query alters retrieved objects.

you must set this flag to True before defining a query.

Query.Mode property

Read-Write See Also

Mode As EQueryType

Sets or returns the query mode, such as kQueryDraw.

property

Туре

iery mode, such as kQueryDraw.

Query.QueryBranch property

Read Only See Also

QueryBranch As QueryBranch

Returns the main branch of the query.

Like all QueryBranch objects, the main branch is a collection of query conditions and sub-branches.To set or change a query branch, use the Query.Define method.

Branch property

QueryBranch

nch of the query.

h objects, the main branch is a collection of query ranches.To set or change a query branch, use the d.

Query.Report property

Read Only See Also

Report As QueryReport

Returns a query report template or an empty object.

Using an empty Query.Report causes an error.

property

leport

rt template or an empty object.

ry.Report causes an error.

Query.ReportDefined property

Read Only See Also

ReportDefined As Boolean

Returns True if the query has a report template associated with it.

Defined property

; Boolean

uery has a report template associated with it.

Query.StringContext property

Read Only See Also

StringContext As String

Returns the string representation of a query.

For example, "Property LAYER=RAILROAD" is a query.

Note To prevent an error, call Query.Define before attempting to use Query.StringContext.

Context property

String

.

presentation of a query. rty LAYER=RAILROAD" is a query. rror, call Query.Define before attempting to use

QueryBranch.Count property

QueryBranch collectionSee AlsoRead Only

Count As Long

Returns the number of sub-branches and leaves in a branch.

Before traversing the query tree, check this property to ensure that it is not empty.

.Count property

See Also

of sub-branches and leaves in a branch.

e query tree, check this property to ensure that it is not

QueryBranch.JoinOp property

QueryBranch collection See Also Read-Write

JoinOP As EJoinOperator

Sets or returns the join operator of the branch.

.JoinOp property

See Also

)perator

in operator of the branch.

QueryBranch.Type property

Read Only See Also

Type As EClassID

Returns kQueryBranch if the object is a branch.

When you get a branch or a condition from a QueryBranch collection, you don't know which it is until you check its Type property. Note that there is a corresponding QueryLeaf.Type property.

The following example builds on sample code for querying objects. For more information, click **2**. Add the following code to the example after qry.Define.This example prints the type of items in the mainqrybr and andqrybr collections.

```
' Define query before running this code
For i = 0 To mainqrybr.Count - 1
Debug.Print mainqrybr.Type
Debug.Print mainqrybr.Item(i).Type
Next i
For i = 0 To andqrybr.Count - 1
Debug.Print andqrybr.Type
Debug.Print andqrybr.Item(i).Type
Next i
```

.Type property

See Also

)

ch if the object is a branch.

condition from a QueryBranch collection, you don't know which it is until you te that there is a corresponding QueryLeaf.Type property.

ple builds on sample code for querying objects. For more Add the following code to the example after nple prints the type of items in the mainqrybr and andqrybr

running this code
'br.Count - 1
'br.Type
'br.Item(i).Type

r.Count - 1 r.Type r.Item(i).Type

QueryCategories.Count property

<u>QueryCategories collection</u> Read Only

Count As Long

Returns the number of query categories in the project.

The following example that traverses the query categories collection prints the name and item number of each category, represented by Queries.Count, and sets up the For loop using QueryCategories.Count.

Dim prj As Project Dim i As Integer For i = 0 To prj.QueryCategories.Count - 1 debug.Print prj.QueryCategories(i).Name debug.Print prj.QueryCategories.Count Next i

ries.Count property

<u>on</u>

of query categories in the project.

ple that traverses the query categories collection prints the per of each category, represented by Queries.Count, and sets 3 QueryCategories.Count.

yCategories.Count - 1 yCategories(i).Name yCategories.Count
QueryCategory.Count property

QueryCategory collection Read Only

Count As Long

Returns the number of queries in the category.

ry.Count property

<u>n</u>

of queries in the category.

QueryCategory.Name property

<u>QueryCategory collection</u> Read-Write

Name As String

Sets or returns the name of the category.

The following example that traverses the query categories collection and prints the name of each category.

Dim prj As Project Dim i As Integer For i = 0 To prj.QueryCategories.Count - 1 debug.Print prj.QueryCategories(i).Name Next i

ry.Name property

me of the category.

<u>n</u>

ple that traverses the query categories collection and prints egory.

yCategories.Count - 1 yCategories(i).Name

QueryLeaf.DataType property

Read Only See Also

DataType As Long

Returns the data type that is the subject of this condition.

Returns either an EPropertyType, ELocationType, or EDataQueryType constant. For example, with a property condition such as "color = red," the data type is kColor. An SQL condition returns –1, because this property does not apply.

ataType property

g

• that is the subject of this condition.

ropertyType, ELocationType, or EDataQueryType constant. property condition such as "color = red," the data type is dition returns –1, because this property does not apply.

QueryLeaf.JoinOp property

QueryLeaf object See Also Read-Write

JoinOp As EJoinOperator

Sets or returns which join operator applies to this condition: Or or And.

inOp property

<u>Also</u>

)perator

ı join operator applies to this condition: Or or And.

QueryLeaf.LocationType property

Read Only See Also

LocationType As EClassID

Returns a location type if this is a location condition, or 1 if it isn't.

Only the following subset of the EClassID constants are valid location types.

kAllBoundary kPointBoundary kCircleBoundary kFenceBoundary kBufferFenceBoundary kPolygonBoundary kWindowBoundary kPolylineBoundary kBufferPolylineBoundary

>cationType property

<u> Also</u>

,

EClassID

pe if this is a location condition, or 1 if it isn't. ubset of the EClassID constants are valid location types.

undary Iry Iry Joundary Boundary

QueryLeaf.Not property

Read-Write See Also

Not As Boolean

Sets or returns whether a Not operator is applied to this condition.

Returns False by default. To apply the Not operator, set it to True.

Note that there is no corresponding QueryBranch.Not property. You can apply a Not operator only to a single condition; you cannot apply it to a group. If you have a situation where applying a Not operator to a group seems to be necessary, you can always manipulate the query structure to avoid it. For example, the expression Not(A Or B) is logically equivalent to Not A and Not B.

ot property

er a Not operator is applied to this condition.

ault. To apply the Not operator, set it to True.

corresponding QueryBranch.Not property. You can apply a a single condition; you cannot apply it to a group. If you re applying a Not operator to a group seems to be necessary, ipulate the query structure to avoid it. For example, the 'B) is logically equivalent to Not A and Not B.

QueryLeaf.Operation property

Read-Write See Also

Operation As EConditionOperator

Sets or returns the comparison operator for a data or property condition.

Attempting to change this property is an error if the leaf does not represent a data or property condition. To understand what part of a condition the Operation property represents, consider the structure of property conditions. They express ideas such as "If an entity's length is greater than 2...." The *greater than* part is given by the property condition's Operation property. The situation with data conditions is analogous.

peration property

onditionOperator

mparison operator for a data or property condition.

e this property is an error if the leaf does not represent a lition. To understand what part of a condition the Operation consider the structure of property conditions. They express entity's length is greater than 2...." The *greater than* part is *y* condition's Operation property. The situation with data bus.

QueryLeaf.Source property

QueryLeaf object See Also Read-Write

Source As String

Sets or returns the source for a data query.

Strings of the following form can be returned.

"Blockname.AttributeTag" "LinkPathName.ColumnName" "ObjectDataTableName.ObjectDataFieldName" "Regapp.EEDFieldName"

The strings represent the attribute tag of a block, column name of an object data table, an object data field name, or an extended entity data field name.

Use this property for data conditions only. If you attempt to set the source for a query that is not of type kDataQueryType, the API does not set the property, and returns the string previously set for a data query or an empty string if none is set. Wild card characters are supported for Blockname.AttributeTag. For information about wild card characters, look up "wild-card characters" on the Index tab of AutoCAD Map UI Help.

ource property

ource for a data query. ing form can be returned. ibuteTag" ColumnName" eName.ObjectDataFieldName" !ldName"

t the attribute tag of a block, column name of an object data field name, or an extended entity data field name.

data conditions only. If you attempt to set the source for a ype kDataQueryType, the API does not set the property, and viously set for a data query or an empty string if none is set. are supported for Blockname.AttributeTag. For information acters, look up "wild-card characters" on the Index tab of lelp.

QueryLeaf.Type property

Read Only See Also

Type As EClassID

Returns the condition type: Property, Data, Location, or SQL.

The following subset of EClassID values is valid.

kQueryCondition kPropertyCondition kDataCondition kLocationCondition kSQLCondition.

When you get a branch or a condition from a QueryBranch collection, you don't know which it is until you check its Type property. Note that there is a corresponding QueryBranch.Type property.

/pe property

<u> Also</u>

)

n type: Property, Data, Location, or SQL.

t of EClassID values is valid.

1

ion

ion

condition from a QueryBranch collection, you don't know which it is until you te that there is a corresponding QueryBranch.Type property.

QueryLeaf.Value property

Read-Write See Also

Value As Variant

Sets or returns a boundary object if this is a location condition, or a string if this is any other type of condition.

To understand what part of a condition the Value property represents, consider the structure of location conditions and property conditions. Location conditions express ideas such as "If an entity is inside some boundary object...." The *boundary object* part is given by the location condition's Value property. Property conditions express ideas such as "If an entity's length is greater than 2...." The *2* part is given by the property condition's Value property. The situation with Data and SQL conditions is analogous.

alue property

<u> Also</u>

ndary object if this is a location condition, or a string if this condition.

part of a condition the Value property represents, consider ion conditions and property conditions. Location conditions s "If an entity is inside some boundary object...." The : is given by the location condition's Value property. express ideas such as "If an entity's length is greater than iven by the property condition's Value property. The nd SQL conditions is analogous.

QueryReport.Count property

QueryReport collection Read Only

QueryReport As Long

Returns the number of expressions in a report template.

Count property

Jong

of expressions in a report template.

QueryReport.IsReportNested property

<u>QueryReport collection</u> Read-Write

IsReportNested As Boolean

Sets or returns if sub-objects are processed.

Set this property to True to include sub-objects in the report. Sub-objects are components of objects with query results. For example, True writes all points in a polygon to output, like objects in a block reference. False writes the first point only.

IsReportNested property

s Boolean

-objects are processed.

True to include sub-objects in the report. Sub-objects are its with query results. For example, True writes all points in like objects in a block reference. False writes the first point
QueryReport.IsTransformEnabled property

QueryReport collection Read-Write

QueryReport As Boolean

Sets or returns if the query evaluates transformed objects.

IsTransformEnabled property

3oolean

query evaluates transformed objects.

QueryReport.ReportFileName property

QueryReport collection Read-Write

ReportFileName As String

Sets or returns the path and file name for the query report.

Note Wherever there is a backslash in the ReportFileName string, you must write two of them. The first one of the pair is an escape character.

ReportFileName property

As String

th and file name for the query report.

e is a backslash in the ReportFileName string, you must he first one of the pair is an escape character.

Range.Expression property

Range object Read-Write

Expression As String

Sets or returns the comparison value of the range's condition, either a literal value or an expression that resolves to such a value.

ssion property

ing

Imparison value of the range's condition, either a literal In that resolves to such a value.

Range.Operation property

Range object See Also Read-Write

Operation As ERangeOperator

Sets or returns the comparitive operator of the range's condition.

tion property

ingeOperator

mparitive operator of the range's condition.

RangeTable.Count property

RangeTable collection Read Only

Count As Long

Returns the number of ranges in the range table.

Count property

of ranges in the range table.

RangeTable.Description property

RangeTable collection Read-Write

Description As String

Sets or returns the description of the range table.

Description property

ring

escription of the range table.

RangeTable.Name property

RangeTable collection Read-Write

Name As String

Sets or returns the name of the range table.

Name property

me of the range table.

RangeTables.Count property

RangeTables collection Read Only

Count As Long

Returns the number of range tables in the collection.

Count property

of range tables in the collection.

SavedQuery.Category property

Read Only See Also

Category As String

Returns the name of the category of the saved query.

Category property

Also

g

the category of the saved query.

SavedQuery.Description property

SavedQuery object Read-Write

Description As String

Sets or returns a description of the saved query.

Description property

ring

cription of the saved query.

SavedQuery.IsExternal property

Read Only See Also

IsExternal As Boolean

Returns True if the saved query resides in an external file.

sExternal property

Also

lean

aved query resides in an external file.

SavedQuery.Name property

SavedQuery object Read-Write

Name As String

Sets or returns the name of the saved query.

Name property

me of the saved query.
SavedQuery.Path property

Read-Write See Also

Path As String

Sets or returns the path and file name if the saved query resides in an external file.

Path property

Also

th and file name if the saved query resides in an external

SaveSet.Count property

Read Only See Also

Count As Long

Returns the number of objects in the save set.

ıt property

<u>Also</u>

of objects in the save set.

SaveSet.CountErased property

Read Only See Also

CountErased As Long

Returns the number of erased objects in the save set.

ntErased property

<u>Also</u>

long

of erased objects in the save set.

SaveSet.CountLocked property

Read Only See Also

CountLocked As Long

Returns the number of locked objects in the save set.

ntLocked property

<u>Also</u>

Long

of locked objects in the save set.

SaveSet.CountNew property

Read Only See Also

CountNew As Long

Returns the number of newly created objects in the save set.

ntNew property

<u>Also</u>

ng

of newly created objects in the save set.

TextDef.Color property

TextDef object Read-Write

Color As String

Sets or returns the text color.

r property

xt color.

TextDef.Height property

TextDef object Read-Write

Height As String

Sets or returns the text height.

ht property

xt height.

TextDef.InsertPoint property

TextDef object Read-Write

InsertPoint As String

Sets or returns the expression for the insertion point of altered text.

Valid values are ".LABELPT" and ".CENTROID".

rtPoint property

ing

apression for the insertion point of altered text. ABELPT" and ".CENTROID".

TextDef.Justification property

TextDef object Read-Write

Justification As String

Sets or returns the alignment of altered text.

The following strings are valid.

"CENTER"
"MIDDLE"
"RIGHT"
"TL"
"TC"
"TR"
"ML"
"MC"
"MR"
"BL"
"BC"
"BR"

fication property

tring

ignment of altered text. s are valid.

TextDef.Layer property

TextDef object Read-Write

Layer As String

Sets or returns the layer where text is drawn.



yer where text is drawn.

TextDef.Property property

Read Only See Also

Property As EAlterationType

The alteration type.

For text alterations, there is only one alteration type, kAlterationTextEntity.

```
The following procedure traverses a collection of property alterations,
distinguishing text definitions from other alterations, and prints the value of the
text definition in the Immediate Window. Call this procedure just before the call
to qry.Define, as shown in the following example. To go to the sample code,
click ≥.
'Call from Altering Queried Objects sample
txtdfprop prj, altls
'Define Query
boolVal = gry.Define(maingrybr)
Sub txtdfprop(prj As Project, altls As AlterLines)
Dim i As Integer
Dim altl As Variant
Dim txtdf As TextDef
For i = 0 To prj.CurrQuery.AlterProp.Count - 1
Set altl = altls.Item(i)
If altl.Property = kAlterationTextEntity Then
Set txtdf = altls.Item(i)
Debug.Print "This is a text definition property # "
& txtdf.Property & " of value " & txtdf.TextValue
Else
Debug.Print "This is not a text definition. It's a property # " & altl.Property
End If
Next i
End Sub
```

TextDef.Rotation property

TextDef object Read-Write

Rotation As String

Sets or returns the text rotation.

tion property

3

xt rotation.

TextDef.TextStyle property

TextDef object Read-Write

TextStyle As String

Sets or returns the text style.
Style property

١g

xt style.

TextDef.TextValue property

Read-Write See Also

TextValue As String

Sets or returns the text for the text alteration.

If you are using a range table to provide strings conditionally, use TextDef.TextValue to reference the range table by supplying a range expression as argument. For more information, click **N**.

The following example shows how to change the text definition in the sample code for altering queried objects. To go to the sample code, click . The sample code sets the text value to the name of the water body. This example changes that text value to the color number of the text. To run the example, add a call to chgtxt just before the call to qry.Define in the sample code, as shown here:

```
'Call from Altering Queried Objects sample
chgtxt prj, altls
'Define Query
boolVal = qry.Define(mainqrybr)
.
.
.
.
Sub chgtxt(prj As Project, altls As AlterLines)
Dim i As Integer
Dim altl As Variant
Dim txtdf As TextDef
For i = 0 To prj.CurrQuery.AlterProp.Count - 1
Set altl = altls.Item(i)
If altl.Property = kAlterationTextEntity Then
Set txtdf = altls.Item(i)
txtdf.TextValue = ".COLOR"
```

End If Next i

End Sub

WindowBound.BottomLeft property

WindowBound object Read-Write

BottomLeft As Point3d

Sets or returns the bottom-left corner of the window boundary.

nd.BottomLeft property

oint3d

ttom-left corner of the window boundary.

WindowBound.UpperRight property

WindowBound object Read-Write

UpperRight As Point3d

Sets or returns the upper-right corner of the window boundary.

id.UpperRight property

oint3d

per-right corner of the window boundary.

EAdeClassId

Class identification codes for data extension objects Common classes

000	kAdeUnknownClass
001	kAdeObj
002	kAdeList
003	kAdeListIterator
004	kAdeResBuf
005	kAdeAlias
006	kAdeAliasList
007	kAdeDir

Drawing index

800	kAdeArray
009	kAdeFloatArray
010	kAdeUnslongArray

- 011 kAdeFloatIndex
- 012 kAdeIndex

Drawing objects

- 013 kAdeDwgObj
- 014 kAdeEntity
- 015 kAdeDwgDictionary

- 016 kAdeQueryEntity
- 017 kAdeCircle
- 018 kAdePoint
- 019 kAdeFence
- 020 kAdeBufferfence
- 021 kAdePolyline
- 022 kAdeBufferPolyline
- 023 kAdeClosedPolyline
- 024 kAdePolygon
- 025 kAdeWindow
- 026 kAdeDwgObjIterator
- 027 kAdeEntityIterator
- 028 kAdeGroupIterator
- 029 kAdeSubentIterator
- 030 kAdeEntHistory
- 031 kAdeDwgDatabase,
- 032 kAdeDwgDatabaseManager
- 033 kAdeDwgDatabaseHolder
- 034 kAdeDwgLock
- 035 kAdeDwg

Drawing set

- 036 kAdeDwgSet
- 037 kAdeDwgSetEntry
- 038 kAdeDwgSetIterator

Current session

- 039 kAdeDwgGroup
- 040 kAdeSession

Feature alteration

- 041 kAdeFeatureAlt
- 042 kAdeGeneralFeat
- 043 kAdeEntityFeat
- 044 kAdeTextFeat
- 045 kAdeHatchFeat

Query definition

- 046 kAdeQueryDefinition
- 047 kAdeQueryLineList
- 048 kAdeBaseQueryLine
- 049 kAdeQueryLine
- 050 kAdeSQLQueryLine
- 051 kAdeQueryUnit

- 052 kAdeLocnQuery
- 053 kAdePropQuery
- 054 kAdeDataQuery
- 055 kAdeSQLBaseQuery
- 056 kAdeSQLCondListQuery
- 057 kAdeSQLOrderByDef
- 058 kAdeSQLStmtQuery
- 059 kAdeQueryAttrs
- 060 kAdeQueryLibrary
- 061 kAdeQueryCategory

Expressions

062	kAdeExpDef
063	kAdeExpEval
064	kAdeExpSQLInfo
065	kAdeRangeTab
066	kAdeRangeTabLine
067	kAdeRangeLib

Coordinate system

068 kAdeCoordSysLibrary

- 069 kAdeCoordSysCategory
- 070 kAdeCoordSysNameDesc
- 071 kAdeCoordSystem
- 072 kAdeDoProjection
- 073 kAdeCSProjFile
- 074 kAdeDatumDesc
- 075 kAdeElipDesc

Template

076	kAdeTemplate
077	kAdeTemplateLine

Preferences

078	kAdeBasePrefDefn
079	kAdeWSpacePrefDefn
080	kAdeWSPrefDefn
081	kAdeKeyViewLine
082	kAdeKeyView
083	kAdeQryPrefDefn
084	kAdeSvBkPrefDefn
085	kAdeExtDbPrefDefn

086 kAdeDbTypePrefDefn

087	kAdeCoordXformPrefDefn
880	kAdeSysPrefDefn
089	kAdeUserProfile
090	kAdeUserList
091	kAdeLog
Utility	
092	kAdeDwgConvt
093	kAdeConfigConvt
094	kAdeProjConvt
095	kAdeSQLinkConvt
096	kAdeDocAccess
097	kAdeDocDefineDoc
098	kAdeDocDefLib
099	kAdeDocDefinition
Temporary	,
100	kAdeSymtbRecord

Topology

- 101 kAdeTopology
- 102 kAdeTopElement
- 103 kAdeTopNode

104	kAdeTopLink
-----	-------------

- 105 kAdeTopPolygon
- 106 kAdeTopVar
- 107 kAdeTopEntity

Validation

- 108 kAdeValidation
- 109 kAdeIRDRecord
- 110 kAdeBaseAbovePrefDefn

EAdeDwgStatus

Attached drawing status codes

- 0 kDwgInactive
- 1 kDwgActive
- 4 kDwgLocked

ıtus

atus codes

ć

EAdeMode

Transactional manager status codes

0	kUnExist	Object is empty.
1	kAdeClosed	Closed object
2	kOpenForWrite	Object was opened for write
3	kOpenForRead	Object was opened for read
4	kOpenForMultiple	Object was open for read several times
5	kErased	Object was erased
6	kShouldBeErased	Object scheduled for deletion

er status codes

Object is empty.

Closed object

- ite Object was opened for write
- ıd Object was opened for read
- ltiple Object was open for read several times

Object was erased

ased Object scheduled for deletion

EAdeNotification

Notifications

- 0 kStartOfAdeCommand
- 1 kEndOfAdeCommand

ition

Command

ommand

EClassId

Object run-time identification codes Location boundaries

01 kLocationBoundary

- 02 kAllBoundary
- 03 kPointBoundary
- 04 kCircleBoundary
- 05 kFenceBoundary
- 06 kBufferFenceBoundary
- 07 kPolygonBoundary
- 08 kWindowBoundary
- 09 kPolylineBoundary
- 10 kBufferPolylineBoundary
- 11 kClosedPolylineBoundary

Query units

12	kQueryUnit	Query unit of undetermined type
13	kQueryCondition	Query condition of undetermined type
14	kLocationCondition	Location condition
15	kPropertyCondition	Property condition
16	kSQLCondition	SQL condition

17	kDataCondition	Data condition
18	kLocationConditionImp	Not intended for public use.
19	kPropertyConditionImp	Not intended for public use.
20	kSQLConditionImp	Not intended for public use.
21	kDataConditionImp	Not intended for public use.
22	kQueryBranch	Query branch, a set of grouped conditions

EConditionOperator

Comparison operators for query conditions.

Note The only valid operator in a string context is kCondEq.

1	kCondEq	Equal
2	kCondGT	Greater than
3	kCondGTorEq	Greater than or equal
4	kCondLT	Less than
5	kCondLTorEq	Less than or equal
6	kCondNotEq	Not equal

perator

rs for query conditions. operator in a string context is kCondEq.

Equal

Greater than

q Greater than or equal

Less than

1 Less than or equal

Not equal

EDataQueryType

Data query types

0	kDataIRD	Internal resources dictionary data tables, object data fields, and an expression
1	kDataAttribute	Blocks and attribute tags
2	kDataLinkTemplate	Link templates and key columns for tables
3	kDataEED	Extended entity data
4	kDataFeature	Feature data

Jype

Internal resources dictionary data tables, object data fields, and an expression

e Blocks and attribute tags

nplate Link templates and key columns for tables

Extended entity data

Feature data
EDataType

Data types for object data fields

- 0 kUnknownType
- 1 kInteger
- 2 kReal
- 3 kCharacter
- 4 kPoint

: data fields

pe

EDwgUpdateStatus

Drawing update status

0	kDwgNonUpdated	Nothing to save
1	kDwgShouldBeReloaded	Lock information updated. Another ADE should only reload the drawing.
4	kDwgShouldBeRequeried	Queried objects have been modified. Another ADE instance should requery.

Status

1S	
dated	Nothing to save
BeReloaded	Lock information updated. Another ADE should only reload the drawing.
BeRequeried	Queried objects have been modified. Another ADE instance should requery.

EErrType

See Also

The following EErrType constants correspond to the AcMap::EErrType enumerators in the ObjectARX API and to the error types in the ADSRX/AutoLISP API. Errors on the error stack are represented by ErrorEntry collections. Error types are returned by the ErrorEntry.ErrType property.

00	kAdeNoMessage	Error condition does not exist.
01	kAdeWarning	ADE (AutoCAD Data Extension) execution warning
02	kAdeError	ADE execution error
03	kAseWarning	ASE (AutoCAD SQL Extension) execution warning
04	kAseError	ASE execution error
05	kAcWarning	AutoCAD execution warning
06	kAcError	AutoCAD execution error
07	kAsiWarning	ASI (AutoCAD SQL Interface) execution warning
08	kAsiError	ASI execution error.
09	kIRDWarning	Extended object data (Xdata) warning
10	kIRDError	Xdata error
13	kMapError	AutoCAD Map execution error
14	kMentorError	
15	kApplicationError	Operating system level application error
16	kDiagMessage	Diagnostic message returned.

EJoinOperator

Join operators for query conditions

- 1 kOperatorAnd
- 2 kOperatorOr

ery conditions

ł

ELocationType

Location condition types

- 0 kLocationInside
- 1 kLocationCrossing

pe	
ypes	
de	

ssing

EPrefType

Option types

- 0 kQryType Query
- 1 kSvBkType Save back
- 2 kCoordXformType Coordinate transformation

Query

Save back

1Type Coordinate transformation

EQueryType

Query modes

- 0 kQueryDraw
- 1 kQueryPreview
- 2 kQueryReport

!W

t

ERangeOperator

Comparison operators for range lines

1	kRangeEq	Equal to
2	kRangeGT	Greater than
3	kRangeGTorEq	Greater than or equal to
4	kRangeLT	Less than
5	kRangeLTorEq	Less than or equal to
6	kRangeNotEq	Not equal to
7	kRangeOtherwise	Otherwise

ator

rs for range lines

Equal to

Greater than

Ξq	Greater than or equal to	
	Less than	
q	Less than or equal to	

- I Not equal to
- wise Otherwise

ESaveQueryOptions

Save query options

1	kSaveDrawingSet	Saves drawing objects
2	kSaveLocationCoordinates	Saves location coordinates
4	kSavePropertyAlteration	Saves property alterations
8	kAutoExecute	Executes a query when it is loaded.

Options

gSet	Saves drawing objects
nCoordinates	Saves location coordinates
yAlteration	Saves property alterations
Ĵ	Executes a query when it is loaded.

ESaveSetObjectType

Types of the objects in the save set

1	kQueriedExisted	Queried objects that are unchanged or were altered by the query
2	kQueriedErased	Queried objects that were deleted
4	kNewlyCreated	New objects added to model space

ectType

in the save set

ted	Queried objects that are unchanged or were
	altered by the query

ed Queried objects that were deleted

ed New objects added to model space

ETableType

Symbol table types

0	kBlockTable	Block names
1	kLayerTable	Layer names
2	kLinetypeTable	Line types
3	kStyleTable	Text styles
4	kRegappTable	AutoCAD regapps
5	kGroupTable	Group names
6	kLinkTemplate	Link template names
7	kODD	Object data table names
8	kMLinestyleTable	Line styles
9	kFeatureTable	Feature names
10	kLineweightTable	Line weights
11	kPlotstyleTable	Plot styles

ole	Block names
ole	Layer names
Table	Line types
le	Text styles
able	AutoCAD regapps
ble	Group names
plate	Link template names
	Object data table names
yleTable	Line styles
able	Feature names
şhtTable	Line weights
Table	Plot styles

EUserRights

User rights

01	kRightsSuperUser	Superuser
02	kRightsAltDwgSet	Can alter drawing set
04	kRightsEditDwg	Can edit drawings
08	kRightsDrawQuery	Can run Draw queries
16	kRightsAltClass	Can edit Feature Class definitions

ıperUser	Superuser
ltDwgSet	Can alter drawing set
litDwg	Can edit drawings
rawQuery	Can run Draw queries
ltClass	Can edit Feature Class definitions
Getting started with VBA

You use the Visual Basic Editor to create macros with objects listed in the Object Browser. You make objects available by referencing type libraries. AutoCAD Map has two libraries, an AutoCAD library and an AutoCAD Map library (exact names vary with released versions).

To reference required libraries

- **1** Open the drawing or project.
- **2** From AutoCAD Map, choose Tools > Macro > Visual Basic Editor.
- **3** Choose Tools > References.
- **4** Locate the AutoCAD Map library in the list of references (exact name varies with released versions), and if it is not checked, click to check it.
- **5** Check that the following required libraries are already checked: the AutoCAD library (exact name varies with released versions), Visual Basic for Applications, OLE Automation, and Microsoft Forms 2.0 Object Library.

Now you can create macros that interact with AutoCAD Map through the Automation API libraries.

d with VBA

asic Editor to create macros with objects listed in the Object objects available by referencing type libraries. AutoCAD s, an AutoCAD library and an AutoCAD Map library (exact eased versions).

aries

; or project.

Map, choose Tools > Macro > Visual Basic Editor.

leferences.

AD Map library in the list of references (exact name varies sions), and if it is not checked, click to check it.

llowing required libraries are already checked: the / (exact name varies with released versions), Visual Basic OLE Automation, and Microsoft Forms 2.0 Object

macros that interact with AutoCAD Map through the aries.

Understanding modules

Modules are windows in the Visual Basic Editor where you write VBA code. The modules have word processing and other features that simplify writing VBA code. There are the following kinds of modules:

Module	Uses
ThisDrawing	Simple procedures to extend the functionality of your drawing or project.
Class	New object definitions
Standard	Procedures and declarations that other modules in your program access frequently
User Form	User interface for your macro

Every VBA project starts with a class module called ThisDrawing. ThisDrawing represents the class object that defines the AutoCAD Map drawing.

For step-by-step information about using ThisDrawing and other modules, click ■.

g modules

's in the Visual Basic Editor where you write VBA code. ord processing and other features that simplify writing VBA ollowing kinds of modules:

J**ses**

imple procedures to extend the functionality of your rawing or project.

ew object definitions

rocedures and declarations that other modules in your rogram access frequently

ser interface for your macro

starts with a class module called ThisDrawing. ThisDrawing object that defines the AutoCAD Map drawing.

rmation about using ThisDrawing and other modules, click

Creating a simple macro

You create and run a VBA macro from either AutoCAD Map or the Visual Basic Editor.

To create and run a macro from AutoCAD Map

- **1** Choose Tools > Macro > Macros.
- **2** Type the name of the macro, for example Hello.
- 3 Click Create.
- 4 Select a project or drawing.

VBA displays a code window and inserts the following statements in it: Sub Hello, and End Sub. By default, the Option Explicit statement is displayed above Sub Hello , to force you to declare variables in a module. If option to include this statement is turned off, you do not see Option Explicit.

- **5** Between the Sub Hello and End Sub statements, type the following code: MsgBox "Hello"
- **6** Choose Run > Run Sub/User Form to run the code. To step through the code, put the cursor between the Sub and End Sub statements, and press F8. To run the code, choose Run > Run Sub.

The Hello message appears in AutoCAD Map.

To create and run a macro from the Visual Basic Editor

- 1 Double-click ThisDrawing in the Project Explorer.
- **2** Choose Insert > Procedure and type the name for the procedure. For example, type Hello.

VBA inserts the following statements: Sub Hello, End Sub, and Option Explicit, which forces you to declare variables in a module if the option to include this statement is turned on.

- **3** Between the Sub Hello and End Sub statements, type the following code: MsgBox "Hello"
- **4** To step through the code, put the cursor between the Sub and End Sub statements, and press F8. To run the code, choose Run > Run Sub.

The Hello message appears in AutoCAD Map.

Saving macros

Using a project to save macros has many advantages, such as easier management of multiple files and restoration of the previous environment. If you save code in a VB or VBA project, you can open the code in the respective IDE only. If you save code in modules, you can import the module into either VB or VBA projects.

You save macros with the following file extensions.

Extension	Description
.dvb	AutoCAD Map project
.bas	Visual Basic
.cls	Class module

For step-by-step information about saving macros, click \blacksquare .

ve macros has many advantages, such as easier management restoration of the previous environment. If you save code in t, you can open the code in the respective IDE only. If you s, you can import the module into either VB or VBA

h the following file extensions.

Description

AutoCAD Map project

Visual Basic

Class module

rmation about saving macros, click **>**.

Working with projects

When you open the Visual Basic Editor, the Project Explorer appears, displaying a hierarchical list of the projects associated with the open AutoCAD Map drawings and projects and the items contained in and referenced by each project. If the Project Explorer is closed, press Ctrl+R to open it.

Use the Project Explorer to navigate your projects, open the code editing window, and view the AutoCAD Map object specified by the code.

You can change the project name, add a description, associate a Help file with the project, list constant declarations, and lock your project so that only you can modify it.

To set project options

- **1** In the Project Explorer, select a project.
- **2** On the Tools menu, click ACADProject Properties.
- **3** Click the appropriate tab, and then enter the information.

ı projects

Visual Basic Editor, the Project Explorer appears, displaying the projects associated with the open AutoCAD Map ts and the items contained in and referenced by each project. er is closed, press Ctrl+R to open it.

orer to navigate your projects, open the code editing e AutoCAD Map object specified by the code.

project name, add a description, associate a Help file with ant declarations, and lock your project so that only you can

olorer, select a project.

u, click ACADProject Properties.

iate tab, and then enter the information.

Using the Object Browser

In the Visual Basic Editor, the Object Browser lists API components alphabetically.

To see API component lists

- **1** From the Visual Basic Editor, choose View > Object Browser.
- **2** From the list of libraries at the top of the Object Browser, select the AutoCAD Map library (exact name varies with released versions).
- **3** In the Classes pane, click a class such as Query. Its properties and methods are listed in the Members pane.
- **4** In the Members pane, click a method or property. Its syntax is displayed in the Details pane below.

ject Browser

Lditor, the Object Browser lists API components

S

Basic Editor, choose View > Object Browser.

braries at the top of the Object Browser, select the ibrary (exact name varies with released versions).

ie, click a class such as Query. Its properties and methods /Iembers pane.

ane, click a method or property. Its syntax is displayed in below.

Inserting files into modules

You can insert class modules, standard modules, and text files that are stored on the hard disk into the active module.

To insert a file

- **1** From AutoCAD Map, choose Tools > Macro > Visual Basic Editor.
- **2** In the Project Explorer, double-click a module to open its code window.
- **3** Click the location in the window where you want to insert the code from the file.
- **4** Choose Insert > File.
- **5** Browse for the folder and file type of the file you want to insert.
- **6** Select the file, and click Open.

The stored code appears in the code window of the active module.

; into modules

modules, standard modules, and text files that are stored on e active module.

Map, choose Tools > Macro > Visual Basic Editor. olorer, double-click a module to open its code window.

'ile.

lder and file type of the file you want to insert.

d click Open.

ears in the code window of the active module.

Finding Help on Microsoft VB and VBA

From AutoCAD Map, choose Tools > Macro > Visual Basic Editor. From Microsoft Visual Basic, choose Help > Microsoft Visual Basic Help.

on Microsoft VB and VBA

p, choose Tools > Macro > Visual Basic Editor. Ial Basic, choose Help > Microsoft Visual Basic Help.

Finding Help on AutoCAD APIs

- **1** From AutoCAD Map, choose Tools > Macro > Visual Basic Editor.
- **2** From Microsoft Visual Basic, choose Help > Microsoft Visual Basic Help
- **3** Click the Contents tab
- 4 Click Shortcut to AutoCAD Map Visual Basic Reference

on AutoCAD APIs

lap, choose Tools > Macro > Visual Basic Editor. sual Basic, choose Help > Microsoft Visual Basic Help tab

AutoCAD Map Visual Basic Reference

Creating object data tables

<u>Step By Step</u> Sample Code The following example creates a table of object data: the AutoCAD database entity name, the color, and the layer of the object. It builds object data records from the table and attaches them to objects in the drawing.

Sub tableproc()

Dim amap As AcadMap Dim ODfdfs As ODFieldDefs Dim ODfdf As ODFieldDef Dim ODtb As ODTable Dim ODrc As ODRecord

Set amap = ThisDrawing.Application. _ GetInterfaceObject("AutoCADMap.Application")

'Create OD Table Definition Set ODfdfs = amap.Projects(ThisDrawing).MapUtil.NewODFieldDefs

' Add Column Headings and Defaults Set ODfdf = ODfdfs.Add("Entity", "Entity name", "", 0) Set ODfdf = ODfdfs.Add("Color", "Object color", acRed, 1) Set ODfdf = ODfdfs.Add("Layer", "Object layer", "0", 2)

'Ensure Table Does Not Exist If amap.Projects(ThisDrawing) .ODTables.Item("SampleOD") Is Nothing Then

'Register OD Table in the drawing Set ODtb = amap.Projects(ThisDrawing) _ .ODTables.Add("SampleOD", "Sample Xdata", ODfdfs, True)

'Create OD Record with Defaults Set ODrc = ODtb.CreateRecord

'Loop Through Entities in Model Space For Each acadObj In ThisDrawing.ModelSpace 'Fill Records with Entity Data ODrc.Item(0).Value = acadObj.EntityName ODrc.Item(1).Value = acadObj.Color ODrc.Item(2).Value = acadObj.Layer

'Attach Record to Entity ODrc.AttachTo(acadObj.ObjectID)

Next

Else

"Table Already Exists MsgBox "Unable to create " & "SampleOD", , _ "Object Data Table Error" End If

End Sub

Altering queried objects using a range table

<u>Step By Step</u> Sample Code The following example defines property alteration for a query. The property alteration uses a range table. To run the example again, clear the query and clear the list of expressions for alterations.

Sub altermap()

Dim amap As AcadMap Dim prj As Project Dim qry As Query Dim maingrybr As QueryBranch Dim grylf As QueryLeaf Dim propgrylf As QueryLeaf Dim datagrylf As QueryLeaf Dim andqrybr As QueryBranch Dim mapu As MapUtil Dim wind As WindowBound Dim boolVal As Boolean Dim atdr As AttachedDrawing Dim dblary As Variant Dim altls As AlterLines Dim altl As Variant Dim txtdf As TextDef Dim rngtb As RangeTable Dim rng As Range Dim altlLay As AlterLine Dim lay As AcadLayer

'Get AutoCAD Map and Project Objects Set amap = ThisDrawing.Application. GetInterfaceObject("AutoCADMap.Application") Set prj = amap.Projects(ThisDrawing)

```
'Get Current Query
Set qry = prj.CurrQuery
qry.Clear
```

'Attach DWG

```
Set atdr = prj.DrawingSet.Add("MAPTUT:\\citymap7.dwg")
```

```
'Get Main Branch
Set mainqrybr = qry.QueryBranch
```

'Create Layout of Leaves and Branch
Set qrylf = mainqrybr.Add(kLocationCondition, kOperatorAnd)
Set andqrybr = mainqrybr.Add(kQueryBranch, kOperatorAnd)

```
Set propqrylf = andqrybr.Add(kPropertyCondition, kOperatorOr)
Set dataqrylf = andqrybr.Add(kDataCondition, kOperatorOr)
```

```
'Get DWG Extents
dblary = prj.DrawingSet.Item("MAPTUT:\\citymap7.dwg").Extents
```

```
'Define Boundary Area for Location
Set mapu = prj.MapUtil
Set wind = mapu.NewWindow( _
mapu.NewPoint3d(dblary(0), dblary(1), 0), _
mapu.NewPoint3d(dblary(2), dblary(3), 0))
```

```
'Specify Draw Query
qry.Mode = kQueryDraw
```

```
'Alter Color and Annotate
Set altls = qry.AlterProp
```

```
altls.RemoveAll
Set altl = altls.Add(kAlterationColor, "RED")
Set txtdf = altls.Add( _
kAlterationTextEntity, _
":NAME@WATER_BODIES")
qry.AlterTag = True
```

'Create MARSH Layer
Set lay = ThisDrawing.Layers.Add("MARSH")

```
'Alter Layer Based on Range Table
prj.RangeTables.Remove ("foulwater")
Set rngtb = prj.RangeTables.Add("foulwater", "shallow water")
Set rng = rngtb.Add("10", "MARSH", kRangeLT)
Set altlLay = prj.CurrQuery.AlterProp.Add( _
kAlterationLayer, _
"(Range :AVG_DEPTH@WATER_BODIES foulwater)")
```

```
'Define Query
boolVal = qry.Define(mainqrybr)
```

```
'Execute Query
boolVal = qry.Execute
ThisDrawing.Application.ZoomExtents
```

```
End Sub
```

AlterLines.Add examples

<u>Simple example</u>
 <u>Range table example</u>

dd examples

<u>ıple</u>
Creating a query

<u>Step By Step</u> Sample Code The following code queries objects from a source file.

Public Sub queryexample()

Dim amap As AcadMap Dim prj As Project Dim qry As Query Dim mainqrybr As QueryBranch Dim qrylf As QueryLeaf Dim propqrylf As QueryLeaf Dim dataqrylf As QueryLeaf Dim andqrybr As QueryBranch Dim mapu As MapUtil Dim wind As WindowBound Dim boolVal As Boolean Dim atdr As AttachedDrawing Dim dblary As Variant

'Get AutoCAD Map and Project Objects Set amap = ThisDrawing.Application. _ GetInterfaceObject("AutoCADMap.Application") Set prj = amap.Projects(ThisDrawing)

'Get Current Query Set qry = prj.CurrQuery qry.Clear

'Attach DWG
Set atdr = prj.DrawingSet.Add("MAPTUT:\\citymap7.dwg")

'Get Main Branch Set mainqrybr = qry.QueryBranch

'Create Layout of Leaves and Branch
Set qrylf = mainqrybr.Add(kLocationCondition, kOperatorAnd)

'Set Main Branch

Set andqrybr = mainqrybr.Add(kQueryBranch, kOperatorAnd)

```
' Set Conditions for Leaves
Set propqrylf = andqrybr.Add(kPropertyCondition, kOperatorOr)
Set dataqrylf = andqrybr.Add(kDataCondition, kOperatorOr)
```

```
'Get DWG Extents
dblary = prj.DrawingSet.Item("MAPTUT:\\citymap7.dwg").Extents
```

```
'Define Boundary Area for Location
Set mapu = prj.MapUtil
Set wind = mapu.NewWindow( _______
mapu.NewPoint3d(dblary(0), dblary(1), 0), ______
mapu.NewPoint3d(dblary(2), dblary(3), 0))
```

```
'Specify Draw Query
qry.Mode = kQueryDraw
```

```
'Define Query
boolVal = qry.Define(mainqrybr)
```

```
'Execute Query
boolVal = qry.Execute
ThisDrawing.Application.ZoomExtents
End Sub
```

Creating object data tables

Step By Step <u>Sample Code</u> The following example creates an object data table and attaches records to drawing objects.

1 Declare variables.

Dim amap As AcadMap Dim acadObj As Object Dim ODfdfs As ODFieldDefs Dim ODfdf As ODFieldDef Dim ODtb As ODTable Dim ODrc As ODRecord

2 Set application and project objects.

Set amap = ThisDrawing.Application. GetInterfaceObject("AutoCADMap.Application")

3 Create the schema for the object data table.

Set ODfdfs = amap.Projects(ThisDrawing).MapUtil.NewODFieldDefs

4 Create fields for object data. For example, create fields for the name, color, and layer of a drawing object. Specify default values, for example, empty string, AcRed, and layer 0, respectively. Add the fields to the table using consecutive index numbers.

```
Set ODfdf = ODfdfs.Add("Entity", "Entity name", "", 0)
Set ODfdf = ODfdfs.Add("Color", "Object color", acRed, 1)
Set ODfdf = ODfdfs.Add("Layer", "Object layer", "0", 2)
```

5 Name the table and test the uniqueness of the name.

If amap.Projects(ThisDrawing).ODTables. Item("SampleOD3") Is Nothing Then

6 Register the OD Table in the drawing and specify the type of object data as Xdata.

Set ODtb = amap.Projects(ThisDrawing) .ODTables.Add("SampleOD3", "Sample Xdata", ODfdfs, True)

7 Create a record of data with defaults specified in step 4.

Set ODrc = ODtb.CreateRecord

8 Loop though each drawing object, get its values for the name, color, and layer of the object, and attach the data to the drawing object.

For Each acadObj In ThisDrawing.ModelSpace ODrc.Item(0).Value = acadObj.EntityName ODrc.Item(1).Value = acadObj.Color ODrc.Item(2).Value = acadObj.Layer ODrc.AttachTo (acadObj.ObjectID) Next

Testing the example

Run the example, and enter ADEEDITDATA at the command prompt to see the output. To run the code again, change the name of the table from SampleOD to another name.

Creating a query

Step By Step <u>Sample Code</u> The following example queries objects from a source file.

Attach source drawings

Attach the source drawings that you will query from. For example, to attach citymap7.dwg, whose path is represented by the drive alias, MAPTUT, use this code.

Dim atdr As AttachedDrawing Set atdr = prj.DrawingSet.Add("MAPTUT:\\citymap7.dwg")

Create the main branch

1 Get the AutoCAD Map application. Dim amap As AcadMap Set amap = ThisDrawing.Application. _ GetInterfaceObject("AutoCADMap.Application")

2 Get the project and the current query.

Dim prj As project Dim qry As Query Set prj = amap.Projects(ThisDrawing) Set qry = prj.CurrQuery

3 Clear the current query's main branch (its definition). Note If you plan to use this query again, save a copy of it in the query library before clearing it. Otherwise, it will be lost.

qry.Clear

4 Get a copy of the main branch, which is now empty.

Dim mainqrybr As QueryBranch Set mainqrybr = qry.QueryBranch

5 Add a location, property, data, or SQL condition to the branch. For example, create a location query to retrieve all objects inside a window.

Dim qrylf As QueryLeaf Set qrylf = mainqrybr.Add(kLocationCondition, kOperatorAnd)

6 Define and set the boundary for the location leaf.

Dim wind As WindowBound Dim boolVal As Boolean Dim mapu As MapUtil Dim dblary As Variant

```
dblary = prj.DrawingSet.Item("MAPTUT:\\citymap7.dwg").Extents
Set mapu = prj.MapUtil
Set wind = mapu.NewWindow( __
mapu.NewPoint3d(dblary(0), dblary(1), 0), __
mapu.NewPoint3d(dblary(2), dblary(3), 0))
boolVal = qrylf.SetLocationCond(kLocationInside, wind)
```

Create sub-branches and leaves

1 Create other leaves and branches. For example, create an AND sub-branch with property and data condition leaves.

Dim andqrybr As QueryBranch Set andqrybr = mainqrybr.Add(kQueryBranch, kOperatorAnd)

2 Set the conditional relationships for the leaves. For example, to the AND sub-branch, add OR leaves.

```
Set propqrylf = andqrybr.Add(kPropertyCondition, kOperatorOr)
Set dataqrylf = andqrybr. .Add(kDataCondition, kOperatorOr)
```

3 Set the property and object data leaves to query objects within a window boundary and that are either on the stream layer or a water body with an average depth of less than 10 feet.

```
boolVal = propqryleaf.SetPropertyCond(kLayer, kCondEq, "Water")
boolVal = dataqryleaf.SetDataCond _
(kDataIRD, kCondLT, "Water_Bodies", "Avg_Depth", 10)
```

4 Redefine the query (assign the branch that you copied and modified as the new main branch).

boolVal = qry.Define(mainbr)

5 Execute the query.

boolVal = qry.Execute