IT10489 - Harnessing the Power of the AutoCAD COM APIs

Lee Ambrosius

Principal Learning Content Developer

Twitter: *http://twitter.com/leeambrosius*

Where Am I and Who Should Be Here

You are in session:
IT10489 - Harnessing the Power of the AutoCAD COM APIs

You should know or have the following to get the most out of this session:

- AutoCAD 2016 (or AutoCAD 2010 and later)
- Visual Studio and VB.NET

Who Am I?

My name is Lee Ambrosius

- Principal Learning Content Developer at Autodesk
- Work on the Customization, Developer, and CAD Administration documentation
- Customizing and programming AutoCAD for 18+ years
- Author of the AutoCAD Customization Platform book series published by Wiley

Session Rules

A few rules for this session:

- Silent your mobile phone and any other device
- If you have to leave at anytime, please do so quietly
- I will allow time to ask questions during the session unless we start getting behind

Thanks for your cooperation

What You Will Learn Today

By the end of this session, you will know how to:

- Link database records to objects in a drawing
- Create and modify sheets and sheet sets
- Check for and fix issues related to drawing standards
- Create a basic transmittal package from a drawing

What You Need Before Getting Started

These should be installed or accessible before starting:

- Microsoft Visual Studio .NET 2012 or 2013
- Windows SDK
- ObjectARX Software Development Kit (SDK)
- AutoCAD Managed .NET Developer’s Guide
- AutoCAD ActiveX Reference Guide

Connectivity Automation Object (CAO) Library

Connecting Objects to a Database

1. Create or open a new Class Library project.
2. Reference the following libraries:
   - *cao20enu.tlb* – Connectivity Automation Object (CAO) library
   - *axdb20enu.tlb* – AutoCAD ActiveX/COM library

Basics of the CAO Library

`DbConnect` is the main object of the library and used to:

- Connect and disconnect from a data source
- Create and modify the label and link templates, and queries stored in a drawing
- Get and attach a link between objects in a drawing to a database record
- Reload the labels of objects in a drawing

Basics of the CAO Library

The following syntax shows how to reference the Database Manager object:

```
' Reference the Database Manager object
Dim oCAO As New CAO.DbConnect
```

Basics of the CAO Library

Connections to a data source are defined in a UDL file.

A UDL file:

- Unicode encoded file
- Contains the database provider and connection strings to connect to the data source

Basics of the CAO Library

Methods used to work with the data connection:

- `Connect()`
- `Disconnect()`
- `IsConnected()`

Defining Link Templates

Link templates define the unique field(s) which will be used to link an object to a database record.

- Stored in a named dictionary; “CONLDEFDictionary”
- Accessible from the `LinkTemplates` collection
- `GetLinkTemplates()` method is used to get the `LinkTemplates` collection
- `Add()` method is used to create a new link template

Linking Objects to a Database Record

Links connect a drawing object to a database record.

A link defines the relationship between:

- A link template
- Values that match those in the fields of a database record

A link is created using the `CreateLink()` method.

Linking Objects to a Database Record

- Links are attached to a drawing object as extended data (Xdata).

```
(-3 ("DCO15" (1071 . 5) (1071 . 5) (1000
. "ProjectsLink") (1004 . "03000000")))
```

- The `GetLinks()` method is used to get the `Links` collection from the objects in a drawing.
- A link can be updated to reference a different database record.

Defining Label Templates

Label templates define the database fields and format of a label.

- Stored in a named dictionary; “CONLABELDictionary”
- Accessible from the `LabelTemplates` collection
- `GetLabelTemplates()` method is used to get the `LabelTemplates` collection
- `Add()` method is used to create a new label template

Attaching Labels to an Object and Creating Freestanding Labels

Labels are leader and Mtext objects that link to a database record.

A label can be

- Attached to an object with a link
- Freestanding and not associated with a linked object

A label is created using the `CreateLabel()` method.

Attaching Labels to an Object and Creating Freestanding Labels

- An attached label is created by passing the `CreateLabel()` method an object ID
- A freestanding label is created by passing the `CreateLabel()` method a coordinate value
- Labels do not reload automatically like fields
- Use the `ReloadLabels()` method to sync the values displayed in a label with the database record

Attaching Labels to an Object and Creating Freestanding Labels

Display of a label can be controlled with these methods:

- `Show()`
- `Hide()`

Creating Queries

Queries are used to view a subset of database records.

- Stored in a named dictionary; “CONQUERYDictionary”
- Accessible from in `Queries` collection
- `Add()` method is used to create a new query
- `GetQueries()` method is used to get the `Queries` collection
- Query strings can be used with API libraries such as DAO and ADO.

Getting Notified of Changes

`DbConnect` events can be used to monitor the:

- Connecting and disconnecting of a data source
- Creation, deletion, and modification of link templates
- Creation, deletion, and modification of label templates
- Creation, deletion, and modification of queries

Getting Notified of Changes

Object events can be used to monitor the:

- Attachment and removal of links
- Creation and deletion of labels

Sheet Set Object (SSO) Library

Taking a Look at the SSO Library

1. Create or open a Class Library project.
2. Reference the following libraries:
   - *acsmcomponents20.tlb* – Sheet Set Object (SSO) library
   - *axdb20enu.tlb* – AutoCAD ActiveX/COM library

Basics of the SSO Library

`AcSmSheetSetMgr` is the main object of the library and used to:

- Open or close a sheet set database
- Create a sheet set
- Find an open sheet set database
- Step through all open sheet set databases
- Register event handlers

Basics of the SSO Library

The following syntax shows how to reference the Sheet Set Manager object:

```
' Reference the Sheet Set Manager object
Dim sheetSetManager As IAcSmSheetSetMgr
sheetSetManager = New AcSmSheetSetMgr
```

Basics of the SSO Library

Other objects important for working with a sheet set:

- `AcSmDatabase` – Used to access the components in a sheet set file
- `AcSmSheetSet` – Reference to the object that contains the subsets and sheets in a sheet set
- `AcSmSubset` – Reference to a subset
- `AcSmSheet` – Reference to a specific drawing layout

Working with a Sheet Set

Before you work with a sheet set, you need know:

- A database must be locked before it can be edited
- After editing, the database needs to be unlocked

Prior to locking or unlocking a database, you should check its current lock status.

Adding Content to the Sheet Set File

- Subsets organize sheets and are represented by `AcSmSubset` objects.
- A subset can be created with the `CreateSubset()` method.
- Sheets are layouts stored in a drawing file and are represented by `AcSmSheet` objects.
- A sheet can be added with the `AddNewSheet()` or `ImportSheet()` method.

Adding Content to the Sheet Set File

These tasks can be performed from a subset

- Add, remove, or import a sheet
- Add and remove a subset
- Step through the sheets in a subset
- Work with the properties of a subset

Adding Content to the Sheet Set File

These tasks can be performed with a sheet

- Change the referencing layout
- Set the plottable status of a sheet
- Work with the properties of a sheet
  - Add new custom properties
  - Modify standard properties; Title, Revision Date, …

Note: After adding a custom sheet property, it needs to be appended to all existing sheets in the sheet set

Other Objects to Know

- `AcSmResources` – Drawing resources
- `AcSmAcDbLayoutReference` – Drawing layout
- `AcSmCustomPropertyBag` – Container for custom sheet and sheet set properties
- `AcSmFileReference` – Resource file, drawing or drawing template

Other Objects to Know

- `AcSmAcDbBlockRecordReference` – Block contained in a drawing or drawing template
- `AcSmCalloutBlocks` – Callout blocks collection
- `AcSmPublishOptions` – Publish options
- `AcSmSheetSelSets` – Sheet set selections collection
- `AcSmSheetSelSet` – Sheet set selection

CAD Standards Plug-ins (CSP) Library

What You Need Before Getting Started

Important: You need to create an Interop assembly of the CAD Standards Type Library.

1. Start the Visual Studio 2012 Command Prompt.
2. Type cd "*c:\ObjectARX 2016\inc-<platform>*" and press Enter.
3. Type *tlbimp acstmgr.tlb* and press Enter.

The *AcStMgr.dll* file will be added to the *inc-<platform>* folder. This only needs to be done once.

Defining a Plug-in

1. Create or open a Class Library project.
2. Reference the following libraries:
   - *acStMgr.tlb* – CAD Standards Manager library
   - *axdb20enu.tlb* – AutoCAD ActiveX/COM library
3. Reference the COM library Microsoft XML Type library (Microsoft XML, v6.0).
4. Check “Make Assembly COM-Visible” under Assembly Information.

Basics of a CAD Standards API

The `IAcStPlugin2` interface is the main object of the library and used to:

- Set the properties for a CAD standard plug-in
- Initialization a plug-in
- Retrieve error and fix objects
- Iterate errors
- Fix and report errors

CAD Standard Properties

Each plug-in must support a standard set of properties and methods that return

- Author name
- Plug-in name and description
- Hyperlink to the documentation for the plug-in
- Icon and version for the plug-in
- Array of object types affected by the plug-in

CAD Standard Properties

These properties and methods must be implemented:

- `Author`
- `Description`
- `HRef`
- `Icon`
- `Name`
- `Version`
- `GetObjectFilter()`

Initialization

A plug-in is initialized during one of the following conditions:

- Opening of a drawing with an associated DWS file, and the plug-in and real-time checking are enabled
- Enabling the plug-in in the CAD Standards dialog box or Batch Standards Checker

Initialization

The following methods must be implemented to initialize a plug-in:

- `Initialize()`
- `SetupForAudit()`

Retrieve Errors and Fixes

After an error has been identified, you can

- Display all possible fixes
- Suggest a recommended fix

Retrieve Errors and Fixes

These methods are used to get the current error and display the differences between the error and fix objects:

- `GetError()`
- `GetAllFixes()`
- `GetRecommendedFix()`
- `GetPropertyDiffs()`

Fix Errors

The `FixError()` method is called when

- An automatic fix is applied
- The user has determined the appropriate fix

Plug-in determines how the error (bad) object should be updated.

Must set the `ResultStatus` property based if the fix was successful or not.

Report Errors

Most reporting is handled by the

- CAD Standards framework
- Batch Standards Checker

The `WritePluginInfo()` method must be updated for the Batch Standards Checker report to be populated with information about the plug-in.

Other Methods to Implement

A plug-in must also implement these methods:

- `CheckSysvar()`
- `StampDatabase()`

The above methods do not seem to be used, but are required.

Load and Use a Plug-in

After a plug-in has been created, it must be registered before it can be used.

Each plug-in must be registered with:

- Windows – Using *RegAsm.exe*
- AutoCAD – Using a key in the Windows Registry

Load and Use a Plug-in

Once registered, a plug-in should be tested under these situations:

- Checking CAD standards from in AutoCAD
- Apply automatic fixes when checking standards in AutoCAD
- Real-time checking in AutoCAD with notifications
- Checking standards with the Batch Standards Checker

Non-Graphical vs. Graphical Plug-in

No fundamental differences

The two samples differ by the following:

- Helper classes in StandardsHelp.vb
- Class name, ProgId, a few of the global variables
- Description() and Name() properties
- GetObjectFilter() and SetupForAudit() methods
- PlugIn_Next() and PlugIn_Clear() methods
- GetAllFixes() and GetRecommendedFix() methods
- FixError() and WriteStandardsItemsInfo() methods

Debugging a Plug-in

Do the following to debug a plug-in from inside AutoCAD:

1. Build and register the debug version of the plug-in with Windows and AutoCAD.
2. In the VS project, specify the location of the AutoCAD EXE to start when debugging begins.
3. Add breakpoints to the code in VS.
4. Add a DWS file to the drawing to be checked and enable the plug-in for checking.
5. Start checking the drawing file.

Autodesk Transmittal Object (ATO) Library

Taking a Look at the ATO Library

1. Create or open a Class Library project.
2. Reference the following libraries:
   - *AcETransmit19.tlb* – Autodesk Transmittal Object (ATO) library
   - *axdb20enu.tlb* – AutoCAD ActiveX/COM library

Basics of the ATO Library

`TransmittalOperation` is the main object of the library and is used to:

- Configure the settings of a transmittal set
- Add drawing and other files to a transmittal set
- Add sheets from a sheet to a transmittal set
- Create a transmittal package from a transmittal set
- Monitor changes to the files in a transmittal set

Basics of the ATO Library

The following syntax shows how to reference the `TransmittalOperation` object:

```
' Create a transmittal operation
Dim tro As TransmittalOperation = _
   New TransmittalOperation()
```

Configuring the Transmittal Interface

The `TransmittalInfo` object is used to control the settings of a transmittal set.

- Specify the destination and other support folders
- Control the inclusion of reference files
- Maintain absolute or use relative paths for references
- Purge unused named styles

Use the `getTransmittalInfoInterface()` method to get the `TransmittalInfo` object.

Adding Files to a Transmittal Set

Any file type can be added to a transmittal set

- Drawing
- Word processor
- Spreadsheet

Use the `addDrawingFile()` method to add drawings.

Other files can be added using the `addFile()` method.

Files referenced by drawing files are automatically added.

Working with Sheet Sets

Sheets from a sheet set can be added to a transmittal set.

You can add:

- One or more sheets can be added with the `addSheets()` method
- The sheets in a sheet selection set can be added with the `addSheetSelset()` method
- All sheets in a sheet set can be added with the `ArchiveSheetSet()` method

Working with Sheet Sets

Before you can add sheets to a transmittal set, you must reference the *AcSmComponents20.tlb* file.

Methods used to add sheets to a transmittal set require access to objects from a sheet set database:

- Sheet objectID (`IAcSmObjectId`)
- Sheet set selection object (`AcSmSheetSelSet`)
- Sheet set object (`AcSmSheetSet`)

Obtaining Information about Files in a Transmittal Set

- Files of a transmittal set are organized in a files graph.
- The files graph is represented by a `TransmittalFilesGraph` object.
- The files graph can be used to identify file dependencies.
- A file in the files graph can be retrieved with the `GetAt()` method.

Obtaining Information about Files in a Transmittal Set

File dependencies are normally managed by adding files to a transmittal set.

File dependencies can be:

- Created with the `Add()` method
- Removed with the `RemoveAt()` method

Creating a Transmittal Package

- A transmittal package is the resulting output created by the transmittal engine from a transmittal set.
- A transmittal package is generated with the `createTransmittalPackage()` method.
- Optionally, a report file can be created and added to a transmittal package.

Creating a Transmittal Package

- The transmittal engine creates a standard transmittal report based on the files in the transmittal set.
- The standard transmittal report can be obtained with the `getTransmittalReport()` method.
- Custom content can be added to the transmittal report using the `addToReport()` method.
- Contents of the transmittal report must be written to a file using an API such as `System.IO.StreamWriter`.

Getting Notified of Changes by the Transmittal Operation

Changes to the files can be monitored by registering event handlers with the transmittal operation.

You can monitor the:

- Addition of files and sheets to a transmittal set
- Processing of drawing files
- Progress of the transmittal package

Archiving and Sending a Transmittal Package

The Transmittal library does not support

- Archiving a transmittal package as a ZIP or EXE file
- Sending an email with a transmittal package as an attachment

Archiving and Sending a Transmittal Package

A ZIP file can be created with:

- `System.IO.Packaging` namespace
- Third-party libraries, such as DotNetZip library

An email can be sent with:

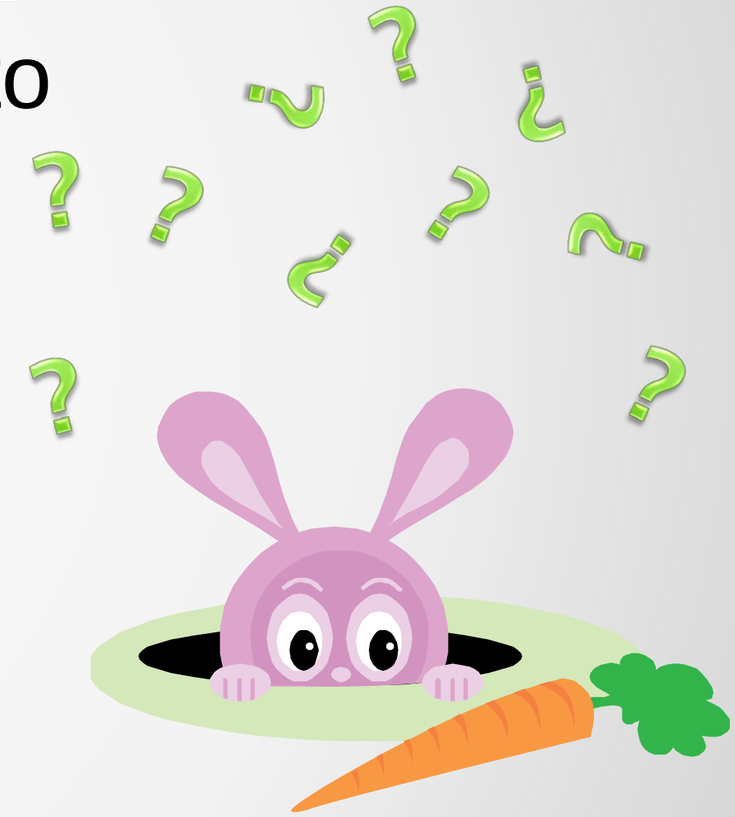
- `System.Net.Mail` namespace
- Third-party libraries, such as Outlook Application library

Final Thoughts and Questions

Final Thoughts and Questions

The ActiveX/COM libraries that AutoCAD supports while are limited can be used to

- Enhance productivity
- Improve or introduce new workflows



Utilizing libraries allows you to open new doors similar to those and the rabbit holes in Lewis Caroll's *Alice's Adventures in Wonderland*.

Closing Remarks

Thanks for choosing this session.

Don’t forget to complete this session’s evaluation.

If you have any further questions, contact me via:

email: *lee.ambrosius@autodesk.com*

twitter: *http://twitter.com/leeambrosius*