**Taking a Look at the Sheet Set Object With VB.NET**

Speaker(s) Lee Ambrosius – Autodesk, Inc.

**CP318-4** The Sheet Set feature was first introduced with AutoCAD® 2005 and is very powerful, but did you know you can create and modify sheet sets without using the Sheet Set Manager? You will learn about working with sheet set files that are open in the Sheet Set Manager, creating a sheet set from scratch, using existing data from an external source to populate properties of a sheet set, and obtaining information from drawings that are in a sheet set. This is not a class on how to use the Sheet Set Manager in AutoCAD, but rather a class that takes a look at working with sheet set files with VB.NET. You will need to know how to work in the VB.NET environment and write macros.

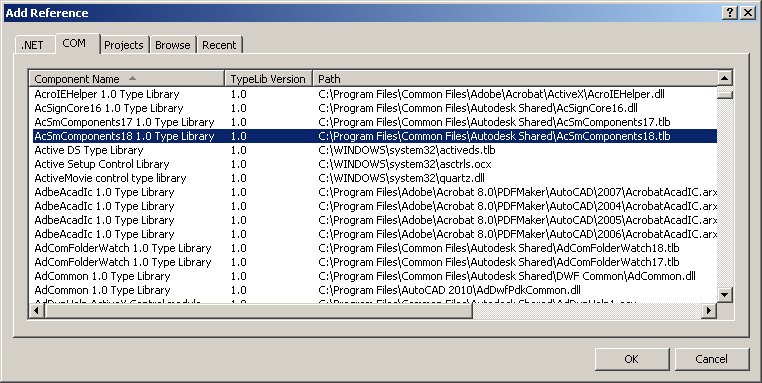
**About the Speaker:**Lee is one of the technical writers on the AutoCAD® team at Autodesk, Inc. and has been an AutoCAD user for 15 years in the fields of architecture and facilities management. He has been teaching AutoCAD users for 10 years at both the corporate and college level. He is best known for his expertise in programming and customizing AutoCAD-based products, and has 10+ years of experience programming with AutoLISP®, VBA, .NET, and ObjectARX®. Lee has written numerous articles for AUGI® and Autodesk. He is the author of AutoCAD 2009 & AutoCAD LT 2009 All-in-One Desk Reference for Dummies and AutoCAD 2008 3D Modeling Workbook for Dummies books

lee.ambrosius@autodesk.com

# 1 Creating a reference to the Sheet Set Object

The Sheet Set object allows you to access the contents of a Sheet Set (DST) file and not the Sheet Set Manager UI. This allows you to create new Sheet Set files, as well as the ability to modify existing Sheet Set files. By being able to access Sheet Set files this way, you can write custom programs that give you more flexibility than just creating a new Sheet Set from scratch or based on an existing one.

The first thing that you need to know is how to reference the AcSmComponents library. The AcSmComponents library can be found in the Autodesk Shared folders. The version of the library that you need is based on the version of AutoCAD you are using. AcSmComponents18 is used for AutoCAD 2010; while AcSmComponents17 is used for AutoCAD 2007 through AutoCAD 2009.



Once you have referenced the library, you need to reference the namespace for the Sheet Set Manager object. To do this, you use the following Imports statement:

Imports ACSMCOMPONENTS18Lib

The Sheet Set Manager object is represented by the object IAcSmSheetSetMgr. The following code demonstrates how to reference the Sheet Set Manager object:

' Get a reference to the Sheet Set Manager object

Dim sheetSetManager As IAcSmSheetSetMgr

sheetSetManager = New AcSmSheetSetMgr

# 2 Taking a look at the Sheet Set Manager Object

Once you have created a reference to the Sheet Set Manager object you can then access any of the sheet set files open in AutoCAD, create a new sheet set file, or open a sheet set file. Below are a few of the commonly used methods available for working with the Sheet Set Manager object and sheet sets.

OpenDatabase – Allows you to open an existing sheet set file.

Close – Allows you to close an open sheet set file.

' Open a Sheet Set

<CommandMethod("ADSK\_OpenSheetSet")> \_

Public Sub OpenSheetSet()

' Get a reference to the Sheet Set Manager object

Dim sheetSetManager As IAcSmSheetSetMgr

sheetSetManager = New AcSmSheetSetMgr

' Open a Sheet Set file

Dim sheetSetDatabase As AcSmDatabase

sheetSetDatabase = sheetSetManager.OpenDatabase("C:\Program Files\AutoCAD

2010\Sample\Sheet Sets\Architectural\IRD Addition.dst", False)

' Return the namd and description of the sheet set

MsgBox("Sheet Set Name: " + sheetSetDatabase.GetSheetSet().GetName() + vbCrLf + \_

"Sheet Set Description: " + sheetSetDatabase.GetSheetSet().GetDesc())

' Close the sheet set

sheetSetManager.Close(sheetSetDatabase)

End Sub

CreateDatabase – Allows you to create a new sheet set file from scratch or based on an existing DST file as a template.

' Create a new sheet set

<CommandMethod("ADSK\_CreateSheetSet")> \_

Public Sub CreateSheetSet()

' Get a reference to the Sheet Set Manager object

Dim sheetSetManager As IAcSmSheetSetMgr

sheetSetManager = New AcSmSheetSetMgr

' Create a new sheet set file

Dim sheetSetDatabase As AcSmDatabase

sheetSetDatabase = sheetSetManager.CreateDatabase("C:\Datasets\CP318-4\

CP318-4.dst", "", True)

' Get the sheet set from the database

Dim sheetSet As AcSmSheetSet

sheetSet = sheetSetDatabase.GetSheetSet()

' Set the name and description of the sheet set

sheetSet.SetName("CP318-4")

sheetSet.SetDesc("AU2009 Sheet Set Object Demo")

' Return the name and description of the sheet set

' Return the namd and description of the sheet set

MsgBox("Sheet Set Name: " + sheetSetDatabase.GetSheetSet().GetName() + vbCrLf + \_

"Sheet Set Description: " + sheetSetDatabase.GetSheetSet().GetDesc())

' Close the sheet set

sheetSetManager.Close(sheetSetDatabase)

End Sub

While the above example code shows the logic of creating a new sheet set and changing its properties, an exception occurs when it executes. The exception occurs because a sheet set must be locked it can be modified. The reason behind it needing to be locked is because more than one user could be working with the sheet set file at the same time. So prior to changing the Name and Description properties, the LockDb method must be called and then UnlockDb must be called to unlock the database once all modifications are made.

The GetLockStatus method is used to return the current lock status of a sheet set database. The returned lock status will be in the form of one of the following constants:

|  |  |
| --- | --- |
| AcSmLockStatus\_UnLocked | Sheet set is not locked, write access denied |
| AcSmLockStatus\_Locked\_Local | Sheet set is locked, write access enabled |
| AcSmLockStatus\_Locked\_Remote | Sheet set is locked by another user, write access denied |
| AcSmLockStatus\_Locked\_ReadOnly | Sheet set is read-only, write access denied |
| AcSmLockStatus\_Locked\_AccessDenied | Write access denied due to lack of user rights |
| AcSmLockStatus\_Locked\_NotConnected | Connection sheet set was lost, write access enabled |

**Note:** The top four statuses are the most commonly encountered out of all possible values.

Below is a custom function that I use to lock and unlock the sheet set database before modifying any of its properties or content.

' Used to lock/unlock a sheet set database

Private Function LockDatabase(ByRef database As AcSmDatabase, \_

ByVal lockFlag As Boolean) As Boolean

Dim dbLock As Boolean = False

' If lockFalg equals True then attempt to lock the database, otherwise

' attempt to unlock it.

If lockFlag = True And \_

database.GetLockStatus() = AcSmLockStatus.AcSmLockStatus\_UnLocked Then

database.LockDb(database)

dbLock = True

ElseIf lockFlag = False And \_

database.GetLockStatus = AcSmLockStatus.AcSmLockStatus\_Locked\_Local Then

database.UnlockDb(database)

dbLock = True

Else

dbLock = False

End If

LockDatabase = dbLock

End Function

To correct the problem with the CreateSheetSet method; you would add the LockDatabase function with the database to be locked and the Boolean value of True before calling the SetName and SetDesc methods. After modifying the properties, you would make a second call to the LockDatabase method to with the Boolean value of False to unlock the database. The code changes would look like:

…

sheetSet = sheetSetDatabase.GetSheetSet()

If LockDatabase(sheetSetDatabase, True) = True Then

' Set the name and description of the sheet set

sheetSet.SetName("CP318-4")

sheetSet.SetDesc("AU2009 Sheet Set Object Demo")

' Unlock the database

LockDatabase(sheetSetDatabase, False)

' Return the name and description of the sheet set

MsgBox("Sheet Set Name: " + sheetSetDatabase.GetSheetSet().GetName() & vbCrLf + \_

"Sheet Set Description: " + sheetSetDatabase.GetSheetSet().GetDesc())

Else

' Display error message

MsgBox("Sheet set could not be opened for write.")

End If

' Close the sheet set

…

GetDatabaseEnumerator – Allows you to step through all open sheet set files in the current AutoCAD session.

' Step through all open sheet sets

<CommandMethod("ADSK\_StepThroughTheOpenSheetSets")> \_

Public Sub StepThroughTheOpenSheetSets()

' Get a reference to the Sheet Set Manager object

Dim sheetSetManager As IAcSmSheetSetMgr

sheetSetManager = New AcSmSheetSetMgr

' Get the loaded databases

Dim enumDatabase As IAcSmEnumDatabase

enumDatabase = sheetSetManager.GetDatabaseEnumerator()

' Get the first open database

Dim item As IAcSmPersist

item = enumDatabase.Next()

Dim customMessage As String = ""

' If a database is open continue

If Not item Is Nothing Then

Dim count As Integer = 0

' Step through the database enumerator

Do While Not item Is Nothing

' Append the file name of the open sheet set to the output string

customMessage = customMessage + vbLf + \_

item.GetDatabase().GetFileName()

' Get the next open database and increment the counter

item = enumDatabase.Next()

count = count + 1

Loop

customMessage = "Sheet sets open: " + count.ToString() + \_

customMessage

Else

customMessage = "No sheet sets are currently open."

End If

' Display the custom message

MsgBox(customMessage)

End Sub

# 3 Adding Content to the Sheet Set File

It is great to be able to open a sheet set file or even create a new one from scratch, but the purpose of a sheet set is to help organize a project. Sheet sets use subsets to help organize the sheets/drawing layouts added to a sheet set file. A subset can represent a physical folder on a network or local drive, or a virtual folder that can be used to just organize the layouts in the sheet set. A subset is represented by the AcSmSubset object. A subset has the following main properties:

* Name (SetName and GetName)
* Description (SetDesc and GetDesc)
* Prompt for Template (SetPromptForDWT and GetPromptForDWT)
* Storage Location for New Sheets (SetNewSheetLocation and GetNewSheetLocation)
* Default Template for New Sheets (SetDefDwtLayout and GetDefDwtLayout)

The following method demonstrates how to create a new subset and set its properties.

' Used to add a subset to a sheet set

Private Function CreateSubset(ByVal sheetSetDatabase As AcSmDatabase, \_

ByVal name As String, \_

ByVal description As String, \_

Optional ByVal newSheetLocation As String = "", \_

Optional ByVal newSheetDWTLocation As String = "", \_

Optional ByVal newSheetDWTLayout As String = "", \_

Optional ByVal promptForDWT As Boolean = False) \_

As AcSmSubset

' Create a subset with the provided name and description

Dim subset As AcSmSubset = sheetSetDatabase.GetSheetSet(). \_

CreateSubset(name, description)

' Get the folder the sheet set is stored in

Dim sheetSetFolder As String

sheetSetFolder = Mid(sheetSetDatabase.GetFileName(), \_

1, InStrRev(sheetSetDatabase.GetFileName(), "\"))

' Create a reference to a File Reference object

Dim fileReference As IAcSmFileReference

fileReference = subset.GetNewSheetLocation()

' Check to see if a path was provided, if not default

' to the location of the sheet set

If newSheetLocation <> "" Then

fileReference.SetFileName(newSheetLocation)

Else

fileReference.SetFileName(sheetSetFolder)

End If

' Set the location for new sheets added to the subset

subset.SetNewSheetLocation(fileReference)

' Create a reference to a Layout Reference object

Dim layoutReference As AcSmAcDbLayoutReference

layoutReference = subset.GetDefDwtLayout

' Check to see that a default DWT location and name was provided

If newSheetDWTLocation <> "" Then

' Set the template location and name of the layout

'for the Layout Reference object

layoutReference.SetFileName(newSheetDWTLocation)

layoutReference.SetName(newSheetDWTLayout)

' Set the Layout Reference for the subset

subset.SetDefDwtLayout(layoutReference)

End If

' Set the Prompt for Template option of the subset

subset.SetPromptForDwt(promptForDWT)

CreateSubset = subset

End Function

Now that the base functionality for creating a subset has been defined, it is time to expand the functionality of the CreateDatabase method.

…

sheetSet.SetName("CP318-4")

sheetSet.SetDesc("AU2009 Sheet Set Object Demo")

' Create two new subsets

Dim subset As AcSmSubset

subset = CreateSubset(sheetSetDatabase, "Plans", "Building Plans", "", \_

"C:\Datasets\CP318-4\CP318-4.dwt", "Sheet", False)

subset = CreateSubset(sheetSetDatabase, "Elevations", "Building Elevations", "", \_

"C:\Datasets\CP318-4\CP318-4.dwt", "Sheet", True)

' Unlock the database

LockDatabase(sheetSetDatabase, False)

…

Now that there are some subsets in the sheet set, you can add sheets to them. You have already seen part of the functionality that is required to add a sheet. This functionality was shown in the creation of a subset. Sheets can exist both as part of a subset or outside of a subset. A sheet in a sheet set is represented by the AcSmSheet object. A sheet has the following main properties:

* Name (SetName and GetName)
* Description (SetDesc and GetDesc)
* Title (SetTitle and GetTitle)
* Number (SetNumber and GetNumber)

The following method demonstrates how to add a new sheet to a sheet set or subset.

' Used to add a sheet to a sheet set or subset

' Note: This function is dependent on a Default Template and Storage location

' being set for the sheet set or subset.

Private Function AddSheet(ByVal component As IAcSmComponent, \_

ByVal name As String, \_

ByVal description As String, \_

ByVal title As String, \_

ByVal number As String) As AcSmSheet

Dim sheet As AcSmSheet

' Check to see if the component is a sheet set or subset,

' and create the new sheet based on the component's type

If component.GetTypeName = "AcSmSubset" Then

Dim subset As AcSmSubset = component

sheet = subset.AddNewSheet(name, description)

' Add the sheet as the first one in the subset

subset.InsertComponent(sheet, Nothing)

Else

sheet = component.GetDatabase().GetSheetSet().AddNewSheet(name, \_

description)

' Add the sheet as the first one in the sheet set

component.GetDatabase().GetSheetSet().InsertComponent(sheet, Nothing)

End If

' Set the number and title of the sheet

sheet.SetNumber(number)

sheet.SetTitle(title)

AddSheet = sheet

End Function

To place the new sheet in the sheet set you use the InsertComponent or InsertComponentAfter methods.

The CreateSheetSet method does not setup a default drawing template or storage location. The process of changing these properties is similar to those used in the CreateSubset method. The following code shows how to set up the default template and storage location.

' Set the default properties of a sheet set

Private Sub SetSheetSetDefaults(ByVal sheetSetDatabase As AcSmDatabase, \_

ByVal name As String, \_

ByVal description As String, \_

Optional ByVal newSheetLocation As String = "", \_

Optional ByVal newSheetDWTLocation As String = "", \_

Optional ByVal newSheetDWTLayout As String = "", \_

Optional ByVal promptForDWT As Boolean = False)

' Set the Name and Description for the sheet set

sheetSetDatabase.GetSheetSet().SetName(name)

sheetSetDatabase.GetSheetSet().SetDesc(description)

' Check to see if a Storage Location was provided

If newSheetLocation <> "" Then

' Get the folder the sheet set is stored in

Dim sheetSetFolder As String

sheetSetFolder = Mid(sheetSetDatabase.GetFileName(), 1, \_

InStrRev(sheetSetDatabase.GetFileName(), "\"))

' Create a reference to a File Reference object

Dim fileReference As IAcSmFileReference

fileReference = sheetSetDatabase.GetSheetSet().GetNewSheetLocation()

' Set the default storage location based on the location of the sheet set

fileReference.SetFileName(sheetSetFolder)

' Set the new Sheet location for the sheet set

sheetSetDatabase.GetSheetSet().SetNewSheetLocation(fileReference)

End If

' Check to see if a Template was provided

If newSheetDWTLocation <> "" Then

' Set the Default Template for the sheet set

Dim layoutReference As AcSmAcDbLayoutReference

layoutReference = sheetSetDatabase.GetSheetSet().GetDefDwtLayout()

' Set the template location and name of the layout

' for the Layout Reference object

layoutReference.SetFileName(newSheetDWTLocation)

layoutReference.SetName(newSheetDWTLayout)

' Set the Layout Reference for the sheet set

sheetSetDatabase.GetSheetSet().SetDefDwtLayout(layoutReference)

End If

' Set the Prompt for Template option of the subset

sheetSetDatabase.GetSheetSet().SetPromptForDwt(promptForDWT)

End Sub

The following code shows the use of the AddSheet and SetSheetSetDefaults methods.

…

' Attempt to lock the database

If LockDatabase(sheetSetDatabase, True) = True Then

' Get the folder the sheet set is stored in

Dim sheetSetFolder As String

sheetSetFolder = Mid(sheetSetDatabase.GetFileName(), 1, \_

InStrRev(sheetSetDatabase.GetFileName(), "\"))

' set the default values of the sheet set

SetSheetSetDefaults(sheetSetDatabase, \_

"CP318-4", "AU2009 Sheet Set Object Demo", \_

sheetSetFolder, \_

"C:\Datasets\CP318-4\CP318-4.dwt", "Sheet")

AddSheet(sheetSetDatabase, "Title Page", "Project Title Page", \_

"Title Page", "T1")

' Create two new subsets

Dim subset As AcSmSubset

subset = CreateSubset(sheetSetDatabase, "Plans", "Building Plans", "", \_

"C:\Datasets\CP318-4\CP318-4.dwt", "Sheet", False)

AddSheet(subset, "North Plan", "Northern section of building plan", \_

"North Plan", "P1")

subset = CreateSubset(sheetSetDatabase, "Elevations", "Building Elevations", \_

"", "C:\Datasets\CP318-4\CP318-4.dwt", "Sheet", True)

' Unlock the database

LockDatabase(sheetSetDatabase, False)

Else

…

It is possible to import an existing layout into a sheet set. The process of importing a sheet versus adding a new sheet is not very different. To import a sheet you can use the ImportSheet or AddSheet method. The following method demonstrates how to import an existing layout from a drawing as a sheet.

' Import a sheet into a sheet set or subset

Private Function ImportASheet(ByVal component As IAcSmComponent, \_

ByVal title As String, \_

ByVal description As String, \_

ByVal number As String, \_

ByVal fileName As String, \_

ByVal layout As String) As AcSmSheet

Dim sheet As AcSmSheet

' Create a reference to a Layout Reference object

Dim layoutReference As New AcSmAcDbLayoutReference

layoutReference.InitNew(component)

' Set the layout and drawing file to use for the sheet

layoutReference.SetFileName(fileName)

layoutReference.SetName(layout)

' Import the sheet into the sheet set

' Check to see if the Component is a Subset or Sheet Set

If component.GetTypeName = "AcSmSubset" Then

Dim subset As AcSmSubset = component

sheet = subset.ImportSheet(layoutReference)

subset.InsertComponent(sheet, Nothing)

Else

Dim sheetSetDatabase As AcSmDatabase = component

sheet = sheetSetDatabase.GetSheetSet().ImportSheet(layoutReference)

sheetSetDatabase.GetSheetSet().InsertComponent(sheet, Nothing)

End If

' Set the properties of the sheet

sheet.SetDesc(description)

sheet.SetTitle(title)

sheet.SetNumber(number)

ImportASheet = sheet

End Function

# 4 Adding Sheet Set and Sheet Properties

Custom properties are a great way to ensure consistency among the values in title blocks for a project, as well as being able to track project status. Custom properties come in two varieties: sheet and sheet set. To create a custom property you need to create a reference to an AcSmCustomPropertyBag object. The AcSmCustomPropertyBag object is used as a container to hold any custom properties that are added to a sheet or sheet set. Once a reference to an AcSmCustomPropertyBag object is made, you can then create a reference to an AcSmCustomPropertyValue object; which is the actual custom property that you want to create.

You use a constant to determine the type of property that will be created. The values that can be used are:

* CUSTOM\_SHEET\_PROP – Sheet property
* CUSTOM\_SHEETSET\_PROP – Sheet set property

The following method demonstrates how to add a new custom property to a sheet or sheet set.

**Note:** It is best to create your properties first before adding any sheets programmatically, otherwise you will need to step through each sheet in the sheet set and add the property to each one manually.

' Set/create a custom sheet or sheet set property

Private Sub SetCustomProperty(ByVal owner As IAcSmPersist, \_

ByVal propertyName As String, \_

ByVal propertyValue As Object, \_

ByVal sheetSetFlag As PropertyFlags)

' Create a reference to the Custom Property Bag

Dim customPropertyBag As AcSmCustomPropertyBag

If owner.GetTypeName() = "AcSmSheet" Then

Dim sheet As AcSmSheet = owner

customPropertyBag = sheet.GetCustomPropertyBag()

Else

Dim sheetSet As AcSmSheetSet = owner

customPropertyBag = sheetSet.GetCustomPropertyBag()

End If

' Create a reference to a Custom Property Value

Dim customPropertyValue As AcSmCustomPropertyValue = New AcSmCustomPropertyValue()

customPropertyValue.InitNew(owner)

' Set the flag for the property

customPropertyValue.SetFlags(sheetSetFlag)

' Set the value for the property

customPropertyValue.SetValue(propertyValue)

' Create the property

customPropertyBag.SetProperty(propertyName, customPropertyValue)

End Sub

The following code shows the use of the SetCustomProperty method.

…

' Set the default values of the sheet set

SetSheetSetDefaults(sheetSetDatabase, \_

"CP318-4", "AU2009 Sheet Set Object Demo", \_

sheetSetFolder, \_

"C:\Datasets\CP318-4\CP318-4.dwt", "Sheet")

' Create a sheet set property

SetCustomProperty(sheetSet, "Project Approved By", "AU09", \_

PropertyFlags.CUSTOM\_SHEETSET\_PROP)

' Create sheet properties

SetCustomProperty(sheetSet, "Checked By", "LAA", \_

PropertyFlags.CUSTOM\_SHEET\_PROP)

SetCustomProperty(sheetSet, "Complete Percentage", "0%", \_

PropertyFlags.CUSTOM\_SHEET\_PROP)

AddSheet(sheetSetDatabase, "Title Page", "Project Title Page", \_

"Title Page", "T1")

…

As previously mentioned, if you create a new sheet property and you have existing sheets in the sheet set it becomes your responsibility to add the properties to the sheets. To add the property to all the sheets in the sheet set you need to step through each of the sheets using a loop. To get all the objects that are contained in the sheet set you use the GetEnumerator method and then use the GetTypeName method to determine which type of object it is you are returned. The following method demonstrates how to step through all the properties associated with a sheet set and make sure the ones that are flagged as a sheet property are added to each sheet in the sheet set.

' Synchronize the properties of a sheet with the sheet set

Private Sub SyncProperties(ByVal sheetSetDatabase As IAcSmDatabase)

' Get the objects in the sheet set

Dim enumerator As IAcSmEnumPersist

enumerator = sheetSetDatabase.GetEnumerator()

' Get the first object in the Enumerator

Dim item As IAcSmPersist

item = enumerator.Next()

' Step through all the objects in the sheet set

Do While Not item Is Nothing

Dim sheet As IAcSmSheet = Nothing

' Check to see if the object is a sheet

If item.GetTypeName() = "AcSmSheet" Then

sheet = item

' Create a reference to the Property Enumerator for

' the Custom Property Bag

Dim enumeratorProperty As IAcSmEnumProperty

enumeratorProperty = item.GetDatabase().GetSheetSet(). \_

GetCustomPropertyBag().GetPropertyEnumerator()

' Get the values from the Sheet Set to populate to the sheets

Dim name As String = ""

Dim customPropertyValue As AcSmCustomPropertyValue = Nothing

' Get the first property

enumeratorProperty.Next(name, customPropertyValue)

' Step through each of the properties

Do While Not customPropertyValue Is Nothing

' Check to see if the property is for a sheet

If customPropertyValue.GetFlags() = \_

PropertyFlags.CUSTOM\_SHEET\_PROP Then

SetCustomProperty(sheet, name, \_

customPropertyValue.GetValue(), \_

customPropertyValue.GetFlags())

End If

' Get the next property

enumeratorProperty.Next(name, customPropertyValue)

Loop

End If

' Get the next Sheet

item = enumerator.Next()

Loop

End Sub

# 5 Working with Sheet Set Events

With the lack of documentation that comes with the Sheet Set Object, it is rather a little difficult to work efficiently with events for the Sheet Set Object. To use an event, it must be register and then unregistered when it is no longer needed. The Sheet Set Object utilizes a single event handler which is designed to return a number of different constant values. Below is an overview of a custom class that implements the events for the Sheet Set Object and two methods used to enable and disable the event handler.

Imports ACSMCOMPONENTS18Lib

' Custom event handler class

Public Class MySSmEventHandler

Implements IAcSmEvents

Private Sub IAcSmEvents\_OnChanged(ByVal ev As AcSmEvent, \_

ByVal comp As IAcSmPersist) \_

Implements IAcSmEvents.OnChanged

Dim activeDocument As Autodesk.AutoCAD.ApplicationServices.Document

activeDocument = Autodesk.AutoCAD.ApplicationServices.Application. \_

DocumentManager.MdiActiveDocument()

Dim oSheet As AcSmSheet

Dim oSubset As AcSmSubset

If ev = AcSmEvent.ACSM\_DATABASE\_OPENED Then

activeDocument.Editor.WriteMessage(vbLf & comp.GetDatabase(). \_

GetFileName() & " was opened.")

End If

If ev = AcSmEvent.ACSM\_DATABASE\_CHANGED Then

activeDocument.Editor.WriteMessage(vbLf & comp.GetDatabase(). \_

GetFileName() & " database changed.")

End If

If ev = AcSmEvent.SHEET\_DELETED Then

oSheet = comp

activeDocument.Editor.WriteMessage(vbLf & oSheet.GetName() & \_

" was deleted.")

End If

If ev = AcSmEvent.SHEET\_SUBSET\_CREATED Then

oSubset = comp

activeDocument.Editor.WriteMessage(vbLf & oSubset.GetName() & \_

" was created.")

End If

If ev = AcSmEvent.SHEET\_SUBSET\_DELETED Then

oSubset = comp

activeDocument.Editor.WriteMessage(vbLf & oSubset.GetName() & \_

" was deleted.")

End If

End Sub

End Class

' Register the event handlers

<CommandMethod("ADSK\_AddSSmEvents")> \_

Public Sub AddSSmEvents()

' Get a reference to the Sheet Set Manager object

Dim sheetSetManager As IAcSmSheetSetMgr

sheetSetManager = New AcSmSheetSetMgr

Try

' Open a Sheet Set file

Dim sheetSetDatabase As IAcSmDatabase

sheetSetDatabase = sheetSetManager. \_

FindOpenDatabase("C:\Datasets\CP318-4\CP318-4.dst")

' Get the sheet set from the database

Dim sheetSet As IAcSmSheetSet

sheetSet = sheetSetDatabase.GetSheetSet()

AddEvents(sheetSetManager, sheetSetDatabase, sheetSet)

Catch ex As Autodesk.AutoCAD.Runtime.Exception

MsgBox(ex.Message, MsgBoxStyle.Critical, "Error")

Catch ex As System.Exception

MsgBox(ex.Message, MsgBoxStyle.Critical, "Error")

End Try

End Sub

' Remove the registered event handlers

<CommandMethod("ADSK\_RemoveSSmEvents")> \_

Public Sub RemoveSSmEvents()

RemoveEvents()

End Sub

' Register event handlers with the Sheet Set Manager,

' database, and sheet set

Private Sub AddEvents(ByVal sheetSetManager As IAcSmSheetSetMgr, \_

ByVal sheetSetDatabase As IAcSmDatabase, \_

ByVal sheetSet As IAcSmSheetSet)

On Error Resume Next

If Not eventHandler Is Nothing Then

Exit Sub

End If

eventHandler = New MySSmEventHandler

m\_sheetSetManager = sheetSetManager

m\_sheetSetDatabase = sheetSetDatabase

m\_sheetSet = sheetSet

' Register an event for the Sheet Set Manager

eventSSMCookie = m\_sheetSetManager.Register(eventHandler)

' Register a database event

eventDbCookie = m\_sheetSetDatabase.Register(eventHandler)

' Register for sheet set event

eventSSetCookie = m\_sheetSet.Register(eventHandler)

End Sub

' Remove the registered event handlers for the Sheet Set Manager,

' database, and sheet set

Private Sub RemoveEvents()

On Error Resume Next

If eventHandler Is Nothing Then

Exit Sub

End If

m\_sheetSetManager.Unregister(eventSSMCookie)

m\_sheetSetDatabase.Unregister(eventDbCookie)

m\_sheetSet.Unregister(eventSSetCookie)

eventHandler = Nothing

End Sub

# 6 Additional Custom Procedures

The following methods demonstrate how to manipulate other properties of a sheet set, and even how to create a custom sheet set property that holds the total number of sheets in the sheet set.

' Counts up the sheets for all the open sheet sets

<CommandMethod("ADSK\_SetSheetCount")> \_

Public Sub SetSheetCount()

Dim nSheetCount As Integer = 0

' Get a reference to the Sheet Set Manager object

Dim sheetSetManager As IAcSmSheetSetMgr

sheetSetManager = New AcSmSheetSetMgr

' Get the loaded databases

Dim enumDatabase As IAcSmEnumDatabase

enumDatabase = sheetSetManager.GetDatabaseEnumerator()

' Get the first open database

Dim item As IAcSmPersist

item = enumDatabase.Next()

Dim sheetSetDatabase As AcSmDatabase

Do While Not item Is Nothing

sheetSetDatabase = item

' Attempt to lock the database

If LockDatabase(sheetSetDatabase, True) = True Then

On Error Resume Next

Dim enumerator As IAcSmEnumPersist

Dim itemSheetSet As IAcSmPersist

' Get the enumerator for the objects in the sheet set

enumerator = sheetSetDatabase.GetEnumerator()

itemSheetSet = enumerator.Next()

' Step through the objects in the sheet set

Do While Not itemSheetSet Is Nothing

' Increment the counter of the object is a sheet

If itemSheetSet.GetTypeName() = "AcSmSheet" Then

nSheetCount = nSheetCount + 1

End If

' Get next object

itemSheetSet = enumerator.Next()

Loop

' Create a sheet set property

SetCustomProperty(sheetSetDatabase.GetSheetSet(), \_

"Total Sheets", CStr(nSheetCount), \_

PropertyFlags.CUSTOM\_SHEETSET\_PROP)

' Unlock the database

LockDatabase(sheetSetDatabase, False)

' Clear and check for the next open sheet set

nSheetCount = 0

Else

MsgBox("Unable to access " & sheetSetDatabase.GetSheetSet().GetName())

End If

item = enumDatabase.Next

Loop

End Sub

The following method demonstrates how to define a sheet selection set.

' Add a sheet selection set

Function AddSheetSelectionSet(ByVal sheetSetDatabase As AcSmDatabase, \_

ByVal name As String, \_

ByVal description As String) As IAcSmSheetSelSet

' Get the sheet selection sets for the sheet set

Dim sheetSelSets As IAcSmSheetSelSets

sheetSelSets = sheetSetDatabase.GetSheetSet().GetSheetSelSets()

' Add the new sheet selection set

Dim sheetSelSet As IAcSmSheetSelSet = Nothing

sheetSelSets.Add(name, description, sheetSelSet)

' Return the selection set added

AddSheetSelectionSet = sheetSelSet

End Function

The following method demonstrates how to setup a resource location.

' Add a drawing resource location to the sheet set

Public Sub AddResourceFileLocation(ByVal sheetSetDatabase As AcSmDatabase, \_

ByVal resourceLoc As String)

Dim resources As IAcSmResources

Dim fileReference As New AcSmFileReference

' Create a reference to a resource location

resources = sheetSetDatabase.GetSheetSet().GetResources()

fileReference.InitNew(sheetSetDatabase)

fileReference.SetFileName(resourceLoc)

' Add the drawing resource location

resources.Add(fileReference)

End Sub

The following method demonstrates how to setup a callout block.

' Add a callout block to the sheet set

Public Sub AddCalloutBlock(ByVal sheetSetDatabase As AcSmDatabase, \_

ByVal blockName As String, \_

ByVal drawingFile As String)

Dim calloutBlocks As IAcSmCalloutBlocks

Dim calloutBlockReference As New AcSmAcDbBlockRecordReference

' Get the callout blocks for the sheet set

calloutBlocks = sheetSetDatabase.GetSheetSet().GetCalloutBlocks()

' Create a new reference to a callout block and define the block to use

calloutBlockReference.InitNew(sheetSetDatabase)

calloutBlockReference.SetFileName(drawingFile)

calloutBlockReference.SetName(blockName)

' Add the new block to the callout blocks for the sheet set

calloutBlocks.Add(calloutBlockReference)

End Sub

The following method demonstrates how to setup a label block.

' Add a label block to the sheet set

Public Sub AddLabelBlock(ByVal sheetSetDatabase As AcSmDatabase, \_

ByVal blockName As String, \_

ByVal drawingFile As String)

Dim labelBlockReference As New AcSmAcDbBlockRecordReference

' Create a reference to a layout label block

labelBlockReference.InitNew(sheetSetDatabase)

labelBlockReference.SetFileName(drawingFile)

labelBlockReference.SetName(blockName)

' Set the label block for the sheet set

sheetSetDatabase.GetSheetSet().SetDefLabelBlk(labelBlockReference)

End Sub

The following method demonstrates how to specify the location of the page setup overrides.

' Add a page setup override to the sheet set

Public Sub AddPageSetupOverride(ByVal sheetSetDatabase As AcSmDatabase, \_

ByVal pageSetupName As String, \_

ByVal drawingFile As String)

Dim fileReference As New AcSmFileReference

' Create a new reference to the page setup overrides to use

fileReference.InitNew(sheetSetDatabase)

fileReference.SetFileName(drawingFile)

' Set the drawing template that contains the page setup overrides

sheetSetDatabase.GetSheetSet().SetAltPageSetups(fileReference)

End Sub

# 7 Where to Get More Information

When you first use a new feature, you will have questions. You need to know where to go and get answers. The following is a list of resources that you can use to get help:

* **Help System** – The topic *Sheet Set Objects Reference* in the AutoCAD Help system contains the documentation related to the Sheet Set Object. The only downside is that the documentation leans heavily on VBA.
* **AUGI Forums** – The AUGI forums provide peer to peer networking where you can ask questions about virtually anything in AutoCAD and get a response from a fellow user. (*http://www.augi.com*)
* **Autodesk Discussion Forums** – The Autodesk forums provide for peer-to-peer networking and some interaction with Autodesk moderators. You can ask a question about anything in AutoCAD and get a response from a fellow user or Autodesk employee. (*http://www.autodesk.com/discussion*)
* **Industry Events and Classes** – Industry events such as AUGI CAD Camp and Autodesk University are great places to learn about new features. Along with industry events, you might also be able to find classes at your local technical college or Autodesk Authorized Training Center (ATC).
* **Autodesk Developer’s Network** – If you are serious about developing applications for AutoCAD, you should consider becoming a registered Autodesk Developer. For information on registering as an Autodesk Developer, see *http://www.autodesk.com/adn*.
* **ObjectARX SDK** – While it is named the ObjectARX SDK, it contains many samples for the Managed .NET API including using the Sheet Set Object with .NET. For information on the ObjectARX SDK, see *http://www.autodesk.com/objectarx*.

**Note:** The sample files used in these handouts can be downloaded from the AU website or *http://www.hyperpics.com/au2009/CP318-4*.

These handouts and others can be downloaded from *http://www.autodesk.com/auonline*.