



Walt Disney World Swan and Dolphin Resort
Orlando, Florida

Taking a Look at the Sheet Set Object.

Lee Ambrosius - HyperPics, LLC

CP15-1 In this course, you will get an understanding of how to access and manipulate sheet sets through the Sheet Set object. This is a great way to write automation for the creation of custom properties or even a sheet set itself. We will explore the many different levels of the Sheet Set object and shows you how to respond to some of the events that are triggered. Sheet sets are a relatively new concept and there are many ways to integrate the process into your company's workflow. By being able to access a sheet set this way, you'll have an additional and effective way to control your company's standards.

About the Speaker:

Lee has been an AutoCAD user for over 10 years in the fields of architecture, facilities management, and mechanical. He has been teaching AutoCAD users for 5+ years at both the corporate and college level. He is best known for his expertise in programming and customizing AutoCAD-based products, and has 8+ years of experience programming with AutoLISP, VBA, .NET, and ObjectARX. Lee writes the "Customization Corner" column for AUGI, and other articles for Autodesk on customization.

lee_ambrosius@hyperpics.com

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 the Sheet Set files this way, you can write custom automation that gives you more flexibility than just creating a new Sheet Set based on an existing one.

The first thing that you need to know is how to reference the AcSmComponents library. The library is limited to early binding, and this limits you to using a programming other than AutoLISP/Visual LISP. This means that you will need to know how to use the VBA editor or another programming language such as VB or VB.NET. The primary focus of this course will be around using VBA from inside of AutoCAD because no additional tools need to be purchased.

Once you have created a reference to the library you need to add the necessary code to reference the Sheet Set Manager Object. The Sheet Set Manager Object is represented by the object AcSmSheetSetMgr. The code below demonstrates how to reference the Sheet Set Manager Object:

" Create a Reference to a Sheet Set Manager Object

```
Dim oSheetSetMgr As AcSmSheetSetMgr
```

```
Set oSheetSetMgr = New AcSmSheetSetMgr
```

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 open Sheet Set files, create a new Sheet Set file or open your own Sheet Set file. Below are a few of the useful methods that are part of the Sheet Set Manager Object.

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

" Open a Sheet Set

```
Public Sub OpenSheetSet()
```

" Create a Reference to the Sheet Set Manager Object

```
Dim oSheetSetMgr As AcSmSheetSetMgr
```

```
Set oSheetSetMgr = New AcSmSheetSetMgr
```

" Open a Sheet Set file

```
Dim oSheetDb As AcSmDatabase
```

```
Set oSheetDb = oSheetSetMgr.OpenDatabase("C:\Program Files\AutoCAD 2006\Sample\Sheet  
Sets\Architectura\NRD Addition.dst", False)
```

" Return the Sheet Set Name and Description

```
MsgBox "Sheet Set Name: " & oSheetDb.GetSheetSet.GetName & vbCrLf + _  
"Sheet Set Description: " & oSheetDb.GetSheetSet.GetDesc
```

" Close the Sheet Set

```
oSheetSetMgr.Close oSheetDb
```

```
End Sub
```

GetDatabaseEnumerator – Allows you to step through all the open sheet set files.

```

" Step through all Open Sheet Sets
Public Sub StepThroughTheSheetSetManager()
  Dim oEnumDb As IAcSmEnumDatabase
  Dim oltem As IAcSmPersist

" Create a Reference to the Sheet Set Manager Object
  Dim oSheetSetMgr As AcSmSheetSetMgr
  Set oSheetSetMgr = New AcSmSheetSetMgr

" Get Loaded Databases
  Set oEnumDb = oSheetSetMgr.GetDatabaseEnumerator

" Get First Open Database
  Set oltem = oEnumDb.Next

" Step through the Databases
  Do While Not oltem Is Nothing

" Display Sheet Set File Name
  MsgBox oltem.GetDatabase.GetFileName

" Get Next Open Database
  Set oltem = oEnumDb.Next
Loop
End Sub

```

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

CreateDatabase – Allows you to create a new sheet set file.

```

" Create a new Sheet Set
Public Sub CreateSheetSet()
" Create a Reference to the Sheet Set Manager Object
  Dim oSheetSetMgr As AcSmSheetSetMgr
  Set oSheetSetMgr = New AcSmSheetSetMgr

" Open a Sheet Set file
  Dim oSheetDb As AcSmDatabase
  Set oSheetDb = oSheetSetMgr.CreateDatabase("C:\Documents and Settings\user name\My Documents\AutoCAD
  Sheet Sets\CP15-1 AU2005.dst", "")

" Set the Name and Description for the Sheet Set
  oSheetDb.GetSheetSet().SetName "CP15-1"
  oSheetDb.GetSheetSet().SetDesc "AU2005 Sheet Set Object Demo for CP15-1"

" Return the Sheet Set Name and Description
  MsgBox "Sheet Set Name: " & oSheetDb.GetSheetSet().GetName & vbCrLf + _
  "Sheet Set Description: " & oSheetDb.GetSheetSet().GetDesc

" Close the Sheet Set
  oSheetSetMgr.Close oSheetDb
End Sub

```

CP15-1: Taking a Look at the Sheet Set Object

The above example of CreateDatabase causes an error to occur and this is due to the Sheet Set not being checked out or locked when updating the Name and Description. Since the Sheet Set can be accessed by multiple users at a single time, the Sheet Set Object supports a file locking mechanism. So before you are allowed to add or make changes to the Sheet Set file, you first must lock the Sheet Set database. The methods for doing this are LockDb and UnlockDb; which are used to lock and unlock the Sheet Set file respectfully.

LockDb – Used to lock the Sheet Set database before making any updates.

UnlockDb – Used to unlock the Sheet Set database after you have made the updates.

To make things easier and to make sure you have permission in the first place to check out the Sheet Set database you use the method GetLockStatus.

GetLockStatus – Used to determine the current lock status of the Sheet Set database. The method will return one of the following constants:

AcSmLockStatus_UnLocked	Write access is denied because the Sheet Set is not locked.
AcSmLockStatus_Locked_Local	Write access is enabled and the Sheet Set is locked.
AcSmLockStatus_Locked_Remote	Write access is denied because another user the Sheet Set locked.
AcSmLockStatus_Locked_ReadOnly	Write access is denied because the Sheet Set is read-only.
AcSmLockStatus_Locked_AccessDenied	Write access is denied due to lack of user rights.
AcSmLockStatus_Locked_NotConnected	Write access is denied because the connection to the Sheet Set was lost.

Note: The top four constants are the most commonly encountered out of the six different possible status values.

Below are two custom functions that I use to lock and unlock the Sheet Set database before modifying any of the content.

" Used to Lock the database (SheetSet)

```
Private Function LockDatabase(oSheetDb As AcSmDatabase) As Boolean
```

" Check the status of the database

```
If oSheetDb.GetLockStatus = AcSmLockStatus_UnLocked Then
```

```
    oSheetDb.LockDb oSheetDb
```

```
    LockDatabase = True
```

```
Else
```

```
    LockDatabase = False
```

```
End If
```

```
End Function
```

" Used to Unlock the database (SheetSet)

```
Private Function UnlockDatabase(oSheetDb As AcSmDatabase) As Boolean
```

" Check the status of the database

```
If oSheetDb.GetLockStatus = AcSmLockStatus_Locked_Local Then
```

```
    oSheetDb.UnlockDb oSheetDb
```

```
    UnlockDatabase = True
```

```
Else
```

```
    UnlockDatabase = False
```

```
End If
```

```
End Function
```

To correct the problem with the CreateDatabase example; you would add the LockDatabase function before calling the SetName and SetDesc methods, and add the UnlockDatabase function after updating the name and description.

```
...  
" Open a Sheet Set file  
Dim oSheetDb As AcSmDatabase  
Set oSheetDb = oSheetSetMgr.CreateDatabase("C:\Documents and Settings\<user name>\My Documents\AutoCAD  
  Sheet Sets\CP15-1 AU2005.dst", "")  
  
" Lock the Database  
LockDatabase oSheetDb  
  
" Set the Name and Description for the Sheet Set  
oSheetDb.GetSheetSet().SetName "CP15-1"  
oSheetDb.GetSheetSet().SetDesc "AU2005 Sheet Set Object Demo for CP15-1"  
  
" Unlock the database  
UnlockDatabase oSheetDb  
  
" Return the Sheet Set Name and Description  
MsgBox "Sheet Set Name: " & oSheetDb.GetSheetSet().GetName & vbCrLf + _  
  "Sheet Set Description: " & oSheetDb.GetSheetSet().GetDesc  
...  
...
```

Other settings that can be set for a Sheet Set:

- Alternative Page Setup (SetAltPageSetups and GetAltPageSetups)
- Callout Blocks (SetCalloutBlocks and GetCalloutBlocks)
- Resource Drawings (SetResources and GetResources)

Adding Content to the Sheet Set File

Its 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. First we will start off by creating a Subset and then add some drawing files from there. A Subset can represent a physical folder on a network or a 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 object AcSmSubset in the Sheet Set 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 procedure demonstrates how to create a new Subset and setup a majority of the necessary options for it.

" Create a Subset in a Sheet Set

```
Private Function CreateSubset(oSheetDb As AcSmDatabase, _  
    strName As String, _  
    strDesc As String, _  
    Optional strNewSheetLocation As String = "", _  
    Optional strNewSheetDWTLLocation As String = "", _  
    Optional strNewSheetDWTLLayout As String = "", _  
    Optional bPromptForDWT As Boolean = False) As AcSmSubset
```

" Create a Subset with the provided name and description

```
Set CreateSubset = oSheetDb.GetSheetSet().CreateSubset(strName, strDesc)
```

" Get the Folder the Sheet Set is Stored in

```
Dim strSheetSetFldr As String  
strSheetSetFldr = Mid(oSheetDb.GetFileName, 1, InStrRev(oSheetDb.GetFileName, "\"))
```

" Create a reference to a File Reference object

```
Dim oFileRef As IAcSmFileReference  
Set oFileRef = CreateSubset.GetNewSheetLocation
```

" Check to see if a path was provided, if not default to the Sheet Set location

```
If strNewSheetLocation <> "" Then  
    oFileRef.SetFileName strNewSheetLocation  
Else  
    oFileRef.SetFileName strSheetSetFldr  
End If
```

" Set the new sheet location for the Subset

```
CreateSubset.SetNewSheetLocation oFileRef
```

" Create a reference to a Layout Reference object

```
Dim oLayoutRef As AcSmAcDbLayoutReference  
Set oLayoutRef = CreateSubset.GetDefDwtLayout
```

" Check to see that a default DWT Location was passed in

```
If strNewSheetDWTLLocation <> "" Then
```

" Set the location of the template in the Layout Reference object

```
oLayoutRef.SetFileName strNewSheetDWTLLocation
```

" Set the Layout name for the Layout Reference object

```
oLayoutRef.SetName strNewSheetDWTLLayout
```

" Set the Layout Reference to the Subset

```
CreateSubset.SetDefDwtLayout oLayoutRef  
End If
```

" Set the Prompt for Template option of the Subset when a new Sheet is created

```
CreateSubset.SetPromptForDwt bPromptForDWT  
End Function
```

Now that the base functionality for creating a Subset has been defined, it is time to expand the functionality of the CreateSheetSet function.

```
...
" Lock the Database
LockDatabase oSheetDb

" Set the Name and Description for the Sheet Set
oSheetDb.GetSheetSet().SetName "CP15-1"
oSheetDb.GetSheetSet().SetDesc "AU2005 Sheet Set Object Demo for CP15-1"

" Create a couple new Subsets
Dim oSubset As AcSmSubset

Set oSubset = CreateSubset(oSheetDb, "Plans", "Building Plans", "", _
    "C:\Documents and Settings\user name\My Documents\AutoCAD Sheet Sets\CP15-1.dwt", _
    "Layout1")

Set oSubset = CreateSubset(oSheetDb, "Elevations", "Building Elevations", "", _
    "C:\Documents and Settings\user name\My Documents\AutoCAD Sheet Sets\CP15-1.dwt", _
    "Layout1")

" Unlock the database
UnlockDatabase oSheetDb
...
```

Now that there are some Subsets in the Sheet Set, lets add some Sheets to it. You have already seen part of the functionality that is required to add a Sheet. This functionality was shown in the creation of the Subset. Sheets can exist outside of a Subset or as part of a Subset based on the level of organization desired. A Sheet in a Sheet Set is represented by the AcSmSheet object.

A Sheet has the following main properties:

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

The following procedure demonstrates how to create a new Sheet in a Sheet Set or Subset based on the default template and storage location.

" Add a Sheet to the Sheet Set or Subset

" This function is dependent on a Default Template and Storage location being setup for the Sheet Set of Subset

```
Private Function AddSheet(oComp As IAcSmComponent, _
    strTitle As String, _
    strDesc As String, _
    strNumber As String) As AcSmSheet
```

" Create a variable to hold a Subset

```
Dim oSubset As AcSmSubset
```

" Add a new Sheet to the Sheet Set

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

```
If oComp.GetTypeName = "AcSmSubset" Then
    Set oSubset = oComp
```

" Create a new Sheet based on the template and location defined by the Subset

```
Set AddSheet = oSubset.AddNewSheet(strTitle, strDesc)
Else
```

" Create a new Sheet based on the template and location defined by the Sheet Set

```
Set AddSheet = oComp.GetDatabase().GetSheetSet().AddNewSheet(strTitle, strDesc)
End If
```

" Add the Title to the Sheet

```
AddSheet.SetTitle strTitle
```

" Add the Number to the Sheet

```
AddSheet.SetNumber strNumber
```

" Add it as the first Sheet

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

```
If oComp.GetTypeName = "AcSmSubset" Then
```

" Add the Sheet to the Subset

```
oSubset.InsertComponent AddSheet, Nothing "oSubset.GetSheetEnumerator().Next
Else
```

" Add the Sheet to the Root of the Sheet Set

```
oComp.GetDatabase().GetSheetSet().InsertComponent AddSheet, Nothing "
    oComp.GetDatabase().GetSheetSet().GetSheetEnumerator().Next
End If
End Function
```

The CreateSheetSet procedure currently doesn't setup a default template and storage location, so to do this you will need to define the correct values for these properties first. The process is very similar to the one that was outlined in the CreateSubset procedure. The code for setting up the default template and storage location is shown in the procedure SetSheetSetDefaults, and the code to add new Sheets to the Sheet Set are defined in the AddSheet procedure. To place the new Sheet in the Sheet set you need to use the method InsertComponent or InsertComponentAfter.

" Setup the Sheet Set Defaults

```
Private Sub SetSheetSetDefaults(oSheetDb As AcSmDatabase, _
    strName As String, _
    strDesc As String, _
    Optional strNewSheetLocation As String = "", _
    Optional strNewSheetDWTLLocation As String = "", _
    Optional strNewSheetDWTLLayout As String = "", _
    Optional bPromptForDWT As Boolean = False)
```

" Set the Name and Description for the Sheet Set

```
oSheetDb.GetSheetSet().SetName strName
oSheetDb.GetSheetSet().SetDesc strDesc
```

" Check to see if a Storage Location was provided

```
If strNewSheetLocation <> "" Then
```

" Get the Folder the Sheet Set is Stored in

```
Dim strSheetSetFldr As String
strSheetSetFldr = Mid(oSheetDb.GetFileName, 1, InStrRev(oSheetDb.GetFileName, "\"))
```

" Create a reference to a File Reference object

```
Dim oFileRef As IAcSmFileReference
Set oFileRef = oSheetDb.GetSheetSet().GetNewSheetLocation
```

" Set the default storage location based on the Sheet Sets location

```
oFileRef.SetFileName strSheetSetFldr
```

" Set the new Sheet location for the Sheet Set

```
oSheetDb.GetSheetSet().SetNewSheetLocation oFileRef
End If
```

" Check to see if a Template was provided

```
If strNewSheetDWTLocation <> "" Then
```

" Add Default Template to Sheet Set

```
Dim oLayoutRef As AcSmAcDbLayoutReference
Set oLayoutRef = oSheetDb.GetSheetSet().GetDefDwtLayout
```

" Set the location of the template in the Layout Reference object

```
oLayoutRef.SetFileName strNewSheetDWTLocation
```

" Set the Layout name for the Layout Reference object

```
oLayoutRef.SetName strNewSheetDWTLayout
```

" Set the Layout Reference to the Subset

```
oSheetDb.GetSheetSet().SetDefDwtLayout oLayoutRef
End If
```

" Set the Prompt for Template option of the Subset when a new Sheet is created

```
oSheetDb.GetSheetSet().SetPromptForDwt bPromptForDWT
End Sub
```

" Add a Sheet to the Sheet Set or Subset

" This function is dependent on a Default Template and Storage location being setup for the Sheet Set of Subset

```
Private Function AddSheet(oComp As IAcSmComponent, _
    strTitle As String, _
    strDesc As String, _
    strNumber As String) As AcSmSheet
```

" Create a variable to hold a Subset

```
Dim oSubset As AcSmSubset
```

" Add a new Sheet to the Sheet Set

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

```
If oComp.GetTypeName = "AcSmSubset" Then
    Set oSubset = oComp
```

" Create a new Sheet based on the template and location defined by the Subset

```
Set AddSheet = oSubset.AddNewSheet(strTitle, strDesc)
Else
```

" Create a new Sheet based on the template and location defined by the Sheet Set

```
Set AddSheet = oComp.GetDatabase().GetSheetSet().AddNewSheet(strTitle, strDesc)
```

End If

" Add the Title to the Sheet

AddSheet.SetTitle strTitle

" Add the Number to the Sheet

AddSheet.SetNumber strNumber

" Add it as the first Sheet

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

If oComp.GetTypeName = "AcSmSubset" Then

" Add the Sheet to the Subset

oSubset.InsertComponent AddSheet, Nothing

Else

" Add the Sheet to the Root of the Sheet Set

oComp.GetDatabase().GetSheetSet().InsertComponent AddSheet, Nothing

End If

End Function

Revised CreateSheetSet procedure that demonstrates how to use the CreateSheetSet procedure.

...

" Lock the Database

LockDatabase sheetdb

" Get the Folder the Sheet Set is Stored in

Dim strSheetSetFldr As String

strSheetSetFldr = Mid(oSheetDb.GetFileName, 1, InStrRev(oSheetDb.GetFileName, "\"))

" Setup the Sheet Set's default values

SetSheetSetDefaults oSheetDb, "CP15-1", "AU2005 Sheet Set Object Demo for CP15-1", _

strSheetSetFldr, _

"C:\Documents and Settings*<user name>*\My Documents\AutoCAD Sheet Sets\CP15-1.dwt", _

"Layout1"

" Create a couple new Subsets

Dim oSubset As AcSmSubset

AddSheet oSheetDb, "Title Page", "Project Title Page", "T1"

Set oSubset = CreateSubset(oSheetDb, "Plans", "Building Plans", "", _

"C:\Documents and Settings*<user name>*\My Documents\AutoCAD Sheet Sets\CP15-1.dwt", _

"Layout1")

AddSheet oSubset, "North Plan", "Northern section of building plan", "P1"

Set oSubset = CreateSubset(oSheetDb, "Elevations", "Building Elevations", "", _

"C:\Documents and Settings*<user name>*\My Documents\AutoCAD Sheet Sets\CP15-1.dwt", _

"Layout1")

" Unlock the database

UnlockDatabase oSheetDb

...

Just like it is possible to add new Sheets through the Sheet Set Manager, you can also import an existing layout into the Sheet Set using the Sheet Set Object. The process of importing a Sheet versus adding a sheet isn't much different. To import a Sheet you use the ImportSheet method which works in a slightly different way from the AddSheet method. However, you can also use the AddSheet method to import a Sheet too.

The following procedure demonstrates how to import an existing layout from a drawing into a Sheet Set.

" Import a Sheet into the Sheet Set or Subset

```
Private Function ImportASheet(oComp As IAcSmComponent, _  
    strTitle As String, _  
    strDesc As String, _  
    strNumber As String, _  
    strFileName As String, _  
    strLayout As String) As AcSmSheet
```

" Create a variable to hold a Subset and Sheet Set

```
Dim oSubset As New AcSmSubset  
Dim oSheetSet As New AcSmDatabase
```

" Create a reference to a Layout Reference object

```
Dim oLayoutRef As New AcSmAcDbLayoutReference  
oLayoutRef.InitNew oComp
```

" Add a new Sheet to the Sheet Set

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

```
If oComp.GetTypeName = "AcSmSubset" Then  
    Set oSubset = oComp
```

" Create a new Sheet based on the template and location defined by the Subset

```
oLayoutRef.SetFileName strFileName  
oLayoutRef.SetName strLayout
```

```
    Set ImportASheet = oSubset.ImportSheet(oLayoutRef)  
Else  
    Set oSheetSet = oComp
```

" Create a new Sheet based on the template and location defined by the Sheet Set

```
oLayoutRef.SetFileName strFileName  
oLayoutRef.SetName strLayout
```

```
    Set ImportASheet = oSheetSet.GetSheetSet().ImportSheet(oLayoutRef)  
End If
```

" Add the Name to the Sheet

```
ImportASheet.SetName strTitle
```

" Add the Description to the Sheet

```
ImportASheet.SetDesc strDesc
```

" Add the Title to the Sheet

```
ImportASheet.SetTitle strTitle
```

" Add the Number to the Sheet

```
ImportASheet.SetNumber strNumber
```

```
" Add it as the first Sheet
```

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

```
If oComp.GetTypeName = "AcSmSubset" Then
```

```
    " Add the Sheet to the Subset
```

```
    oSubset.InsertComponent ImportASheet, Nothing
```

```
Else
```

```
    " Add the Sheet to the Root of the Sheet Set
```

```
    oSheetSet.GetSheetSet().InsertComponent ImportASheet, Nothing
```

```
End If
```

```
End Function
```

Adding Sheet Set and Sheet Properties

Custom properties are a great way to ensure consistency among the values in title blocks for the 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 what is known as a Custom Property Bag (AcSmCustomPropertyBag). The Custom Property Bag is used as a container to hold any custom properties that are added to a Sheet or Sheet Set. Once you have a reference to the Custom Property Bag you then need to create a reference to a Custom Property Value (AcSmCustomPropertyValue); which is the object that is used to reference the object that is added to the Custom Property Bag.

A flag for the Custom Property Value is used to determine if it is a property at the Sheet or Sheet Set level. The value used to refer to a Sheet property is CUSTOM_SHEET_PROP and the value used to refer to a Sheet Set value is CUSTOM_SHEETSET_PROP. The following procedure demonstrates how to create a reference to the Custom Property Bag and add a new property to it.

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. Not that hard to do as you will see in a little while, but if it can be avoided; your program will run more efficiently.

```
" Set/Create a Sheet Set Property
```

```
Private Sub SetCustomProperty(oSheetDb As AcSmDatabase, _
    strName As String, _
    strValue As Variant, _
    Optional bSheetSetFlag As Boolean = True)
```

```
" Create a Reference to the Custom Property Bag
```

```
Dim cBag As AcSmCustomPropertyBag
```

```
Set cBag = oSheetDb.GetSheetSet().GetCustomPropertyBag
```

```
" Create a Reference to a Custom Property Value
```

```
Dim cBagVal As New AcSmCustomPropertyValue
```

```
cBagVal.InitNew oSheetDb.GetSheetSet() " cBag
```

```
" Set the Flag for Sheet Set or Sheet Property
```

```
If bSheetSetFlag = True Then
```

```
    cBagVal.SetFlags CUSTOM_SHEETSET_PROP
```

```
Else
```

```
    cBagVal.SetFlags CUSTOM_SHEET_PROP
```

```
End If
```

```
" Set the value for the Bag
```

```
cBagVal.SetValue strValue
```

" Create the property

```
cBag.SetProperty strName, cBagVal
```

" Cleat variable

```
Set cBagVal = Nothing
End Sub
```

A portion of the revised CreateSheetSet procedure demonstrating how to use the SetCustomPropety procedure.

```
...
" Setup the Sheet Set's default values
SetSheetSetDefaults oSheetDb, "CP15-1", "AU2005 Sheet Set Object Demo for CP15-1", _
    strSheetSetFldr, _
    "C:\Documents and Settings\user name\My Documents\AutoCAD Sheet Sets\CP15-1.dwt", _
    "Layout1"

" Create a Sheet Set Property
SetCustomProperty oSheetDb, "Project Approved By", "AU05"

" Create a Sheet Property
SetCustomProperty oSheetDb, "Checked By", "LAA", False
SetCustomProperty oSheetDb, "Complete Percentage", "0%", False

" Create a couple new Subsets
Dim oSubset As AcSmSubset
...
```

As previously mentioned, if you create a new Sheet property for existing Sheets you become responsible for it. 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 procedure 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 is added to each Sheet in the Sheet Set.

" Synchronize Sheets with Sheet Properties

```
Private Sub SyncSheetProperties()
" Create a Reference to the Sheet Set Manager Object
Dim oSheetSetMgr As AcSmSheetSetMgr
Set oSheetSetMgr = New AcSmSheetSetMgr

" Get the current Sheet Set
Dim oSheetDb As AcSmDatabase
Set oSheetDb = oSheetSetMgr.GetDatabaseEnumerator().Next

" Lock the Database
If LockDatabase(oSheetDb) Then

" Get the objects in the Sheet Set
Dim oEnum As IAcSmEnumPersist
Set oEnum = oSheetDb.GetEnumerator

" Get the first object in the Enumerator
```

```
Dim oltem As IAcSmPersist
Set oltem = oEnum.Next
```

" Step through all the objects in the Sheet Set

```
Do While Not oltem Is Nothing
```

" Check to see if the object is a Sheet

```
If oltem.GetTypeName = "AcSmSheet" Then
```

```
Dim oSheet As AcSmSheet
```

```
Set oSheet = oltem
```

" Create a reference to the Property Enumerator for the Custom Property Bag

```
Dim oEnumProp As IAcSmEnumProperty
```

```
Set oEnumProp = oSheet.GetDatabase().GetSheetSet().GetCustomPropertyBag().GetPropertyEnumerator
```

" Get the values from the Sheet Set to populate to the Sheets

```
Dim strName As String
```

```
Dim oPropVal As AcSmCustomPropertyValue
```

" Get the first property

```
oEnumProp.Next strName, oPropVal
```

" Step through each of the properties

```
Do While Not oPropVal Is Nothing
```

" Check to see if the Property is for a Sheet and if so continue

```
If oPropVal.GetFlags() = CUSTOM_SHEET_PROP Then
```

" Create a reference to the Custom Property Bag

```
Dim cBag As AcSmCustomPropertyBag
```

```
Dim cBagVal As New AcSmCustomPropertyValue
```

" Create a reference to the Custom Property Value

```
Set cBag = oSheet.GetCustomPropertyBag
```

" Create a new Custom Property Value

```
cBagVal.InitNew cBag
```

" Set the Property Flag to a Sheet Property

```
cBagVal.SetFlags CUSTOM_SHEET_PROP
```

" Set the Value for the Property

```
cBagVal.SetValue oPropVal.GetValue
```

" Set the Name for the Property

```
cBag.SetProperty strName, cBagVal
```

```
Set cBagVal = Nothing
```

```
End If
```

" Get the next Property

```
oEnumProp.Next strName, oPropVal
```

```
Loop
```

```
End If
```

" Get the next Sheet

```
Set oltem = oEnum.Next
```



```

Loop

" Unlock the database
UnlockDatabase oSheetDb
Else
MsgBox "Unable to access "" & sheetdb.GetSheetSet().GetName & "" Sheet Set."
End If
End Sub

```

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. Events for the Sheet Set Object don't work like they do with the AutoCAD ActiveX (COM) API, or for that matter many of the other APIs that are commonly used with VBA . You need to register and unregister the events yourself, which can make it hard to understand and implement. There is 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 procedures used to enable and disable the event handler.

" Begin clsEventHandler module (Class module)

Implements IAcSmEvents

" Custom Class to handle Events for the Sheet Set Object

```

Private Sub IAcSmEvents_OnChanged(ByVal ev As AcSmEvent, ByVal comp As IAcSmPersist)
Dim oSheet As AcSmSheet
Dim oSubset As AcSmSubset

If ev = ACSM_DATABASE_OPENED Then
ThisDrawing.Utility.Prompt vbLf & comp.GetDatabase.GetFileName & " was opened."
ElseIf ev = ACSM_DATABASE_CHANGED Then
ThisDrawing.Utility.Prompt vbLf & "database changed"
ElseIf ev = SHEET_DELETED Then
Set oSheet = comp
ThisDrawing.Utility.Prompt vbLf & oSheet.GetName & " was deleted"
ElseIf ev = SHEET_SUBSET_CREATED Then
Set oSubset = comp
ThisDrawing.Utility.Prompt vbLf & oSubset.GetName & " was created"
ElseIf ev = SHEET_SUBSET_DELETED Then
Set oSubset = comp
ThisDrawing.Utility.Prompt vbLf & oSubset.GetName & " was deleted"
End If
End Sub

```

" Begin basEventsSample module

```

Option Explicit
Dim oSheetSetMgr As IAcSmSheetSetMgr
Dim oSheetDb As IAcSmDatabase
Dim oSheetSet As IAcSmSheetSet

Dim eHndlr As clsEventHandler
Dim eSSMCookie As Long
Dim eDbCookie As Long
Dim eSSetCookie As Long

```

" Test procedure for working with Events

Public Sub TestEvents()

" Create a reference to the custom EventHandler Class

Set eHndlr = New clsEventHandler

" Create a Reference to the Sheet Set Manager Object

Set oSheetSetMgr = New AcSmSheetSetMgr

" Open a Sheet Set file

Set oSheetDb = oSheetSetMgr.FindOpenDatabase("C:\Program Files\AutoCAD 2006\Sample\Sheet Sets\Architectural\NRD Addition.dst")

" Register the event handlers

eSSMCookie = oSheetSetMgr.Register(eHndlr)

eDbCookie = oSheetDb.Register(eHndlr)

Set oSheetSet = oSheetDb.GetSheetSet

eSSetCookie = oSheetSet.Register(eHndlr)

End Sub

" Unregister Events

Public Sub EndTestEvents()

oSheetSetMgr.Unregister eSSMCookie

oSheetDb.Unregister eDbCookie

oSheetSet.Unregister eSSetCookie

Set eHndlr = Nothing

End Sub

Additional Custom Procedures

The following procedure demonstrates how to step through and count up all of the Sheets found in a Sheet Set, and create a custom Sheet Set property that will hold the total number of sheets.

" Counts up the Sheets for all the Open Sheet Sets

Public Sub SetSheetCount()

Dim nSheetCount As Integer

Dim oEnumDb As IAcSmEnumDatabase

Dim oltem As IAcSmPersist

" Create a Reference to the Sheet Set Manager Object

Dim oSheetSetMgr As AcSmSheetSetMgr

Set oSheetSetMgr = New AcSmSheetSetMgr

Set oEnumDb = oSheetSetMgr.GetDatabaseEnumerator

Set oltem = oEnumDb.Next

Dim oSheetDb As AcSmDatabase

Do While Not oltem Is Nothing

Set oSheetDb = oltem

" Lock the Database

If LockDatabase(oSheetDb) Then

```
On Error Resume Next

Dim oEnum As IAcSmEnumPersist
Dim oltemSh As IAcSmPersist

" Get the Enumerator for the objects in the Sheet Set
Set oEnum = oSheetDb.GetEnumerator
Set oltemSh = oEnum.Next

" Step through the objects in the Sheet Set
Do While Not oltemSh Is Nothing
  " Increment the counter of the object is a Sheet
  If oltemSh.GetTypeName = "AcSmSheet" Then
    nSheetCount = nSheetCount + 1
  End If

  " Get next object
  Set oltemSh = oEnum.Next
Loop

" Apply the Sheet Count as a custom property
Dim cBag As AcSmCustomPropertyBag
Dim cBagVal As New AcSmCustomPropertyValue

Set cBag = oSheetDb.GetSheetSet().GetCustomPropertyBag

cBagVal.InitNew cBag

cBagVal.SetFlags CUSTOM_SHEETSET_PROP
cBagVal.SetValue CStr(nSheetCount)

cBag.SetProperty "Total Sheets", cBagVal

Set cBagVal = Nothing

" Unlock the database
UnlockDatabase oSheetDb

" Clear and check for next SheetSet that is open
nSheetCount = 0
Else
  MsgBox "Unable to access """" & sheetdb.GetSheetSet().GetName & """" Sheet Set."
End If

Set oltem = oEnumDb.Next
Loop
End Sub
```