Automating Plots in ArcMap: Tips and Tricks

Mark Cederholm

UniSource Energy Services

Printing a Layout

- Getting the printers and pages
- Setting the printer
- Setting the printer paper
- Scaling the layout to the printer
- Sending the layout to the printer

Getting the printers

```
Dim pPN As IEnumPrinterNames

cbSelectPrinter.Clear
Set pPN = Application
pPN.Reset
sVal = pPN.Next
Do While sVal <> ""
   cbSelectPrinter.AddItem sVal
   sVal = pPN.Next
Loop
If cbSelectPrinter.ListCount = 0 Then
   MsgBox "WARNING:  No printers defined.", vbExclamation
   Exit Sub
End If
cbSelectPrinter.ListIndex = 0
```

Setting the printer

```
Dim sVal As String
Dim pApp As IMxApplication
Dim pPaper As IPaper

If cbSelectPrinter.ListCount = 0 Then
   Exit Sub
End If

sVal = cbSelectPrinter.Value
Set pApp = Application
Set pPaper = pApp.Paper
pPaper.PrinterName = sVal
```

Getting the printer pages

```
Dim pForms As IEnumNamedID
Dim iFormID As Long
Dim sFormName As String

cbSelectPage.Clear
Set pForms = pPaper.Forms
pForms.Reset
iFormID = pForms.Next(sFormName)
Do While sFormName <> ""
   cbSelectPage.AddItem sFormName
   iFormID = pForms.Next(sFormName)
Loop

cbSelectPage.ListIndex = 0
```

Setting the printer paper

```
Dim sVal As String
Dim pApp As IMxApplication
Dim pPaper As IPaper
Dim pForms As IEnumNamedID
Dim iFormID As Long
Dim sFormName As String

sVal = cbSelectPage.Value
Set pApp = Application
Set pPaper = pApp.Paper
pPaper.Orientation = 2  ' landscape
Set pForms = pPaper.Forms
pForms.Reset
iFormID = pForms.Next(sFormName)
Do While sFormName <> ""
   If sFormName = sVal Then
      Exit Do
   End If
   iFormID = pForms.Next(sFormName)
Loop
pPaper.FormID = iFormID
```

Scale the layout to the printer paper (instead of changing the page size!)

```
Dim pDoc As IMxDocument
Dim pLayout As IPageLayout
Dim pPage As IPage

Set pDoc = Application.Document
Set pLayout = pDoc.PageLayout
Set pPage = pLayout.Page
pPage.PageToPrinterMapping = esriPageMappingScale
```

Printing the layout

```
Dim pPrinter As IPrinter
Dim pPrinterBounds As IEnvelope
Dim deviceRECT As tagRECT

Set pPrinter = pApp.Printer
Set pPrinterBounds = New Envelope
pPage.GetDeviceBounds pPrinter, 1, 0, pPrinter.Resolution, pPrinterBounds
With deviceRECT
   .bottom = Round(pPrinterBounds.YMax)
   .Left = Round(pPrinterBounds.XMin)
   .Right = Round(pPrinterBounds.XMax)
   .Top = Round(pPrinterBounds.YMin)
End With
```

Printing the layout (continued)

```
Dim pVisibleBounds As IEnvelope
Dim w As Double, h As Double

Set pVisibleBounds = New Envelope
pPage.QuerySize w, h
pVisibleBounds.XMin = 0
pVisibleBounds.YMin = 0
pVisibleBounds.XMax = w
pVisibleBounds.YMax = h

Dim pActiveView As IActiveView
Dim hDC As Long
Dim pCancel As ITrackCancel

Set pActiveView = pLayout
hDC = pPrinter.StartPrinting(pPrinterBounds, 0)
Set pCancel = New CancelTracker
pActiveView.Output hDC, pPrinter.Resolution, deviceRECT, pVisibleBounds, pCancel
pPrinter.FinishPrinting
```

Tips and Tricks

Storing a TOC in a table

```
Dim pLayer As ILayer
Dim pRow As iRow
Dim pMemBlobStream As IMemoryBlobStream
Dim pObjectStream As IObjectStream
Dim pPropset As IPropertySet
Dim pPersistStream As IPersistStream

iNumLayers = pMap.LayerCount
For i = 0 To iNumLayers - 1
  Set pLayer = pMap.Layer(i)
  Set pRow = pTable.CreateRow
  …
  pRow.Value(iLayerIndex) = i
  Set pMemBlobStream = New MemoryBlobStream
  Set pObjectStream = New ObjectStream
  Set pObjectStream.Stream = pMemBlobStream
  Set pPropset = New PropertySet
  Set pPersistStream = pPropset
  pPropset.SetProperty "Layer", pLayer
  pPersistStream.Save pObjectStream, False
  pRow.Value(iData) = pMemBlobStream
  pRow.Store
Next
```

Restoring a TOC: sort layers in reverse

```
Dim pDT As IDisplayTransformation
Dim pViewExtent As IEnvelope

Set pDT = pActiveView.ScreenDisplay.DisplayTransformation
Set pViewExtent = pDT.VisibleBounds
pMap.ClearLayers

Dim pTS As ITableSort
Dim pCursor As ICursor
Dim pRow As iRow

Set pTS = New TableSort
With pTS
   .Fields = "layer_index"
   Set .Table = pTable
   .Ascending("layer_index") = False
   Set .QueryFilter = pQF
End With
pTS.Sort Nothing
Set pCursor = pTS.Rows
Set pRow = pCursor.NextRow
```

Restoring a TOC: add layers

```
Dim pMemBlobStream As IMemoryBlobStream
Dim pObjectStream As IObjectStream
Dim pPropset As IPropertySet
Dim pPersistStream As IPersistStream
Dim pLayer As ILayer

iData = pTable.FindField("layer_data")
Do While Not pRow Is Nothing
   Set pMemBlobStream = pRow.Value(iData)
   Set pObjectStream = New ObjectStream
   Set pObjectStream.Stream = pMemBlobStream
   Set pPropset = New PropertySet
   Set pPersistStream = pPropset
   pPersistStream.Load pObjectStream
   Set pLayer = pPropset.GetProperty("Layer")
   pMap.AddLayer pLayer
   Set pRow = pCursor.NextRow
Loop
pDT.VisibleBounds = pViewExtent
pDoc.UpdateContents
```

Loading a layout template

```
Dim pDoc As IMxDocument
Dim pGxFile As IGxFile
Dim pGxPageLayout As IGxMapPageLayout
Dim pPageLayout As IPageLayout

Set pDoc = ThisDocument
Set pGxFile = New GxMap
pGxFile.Path = sTemplatePath
Set pGxPageLayout = pGxFile
Set pPageLayout = pGxPageLayout.PageLayout
pPageLayout.ReplaceMaps pDoc.Maps
Set pDoc.PageLayout = pPageLayout

Dim pPage As IPage
Dim pApp As IMxApplication

Set pPage = pPageLayout.Page
pPage.PageToPrinterMapping = esriPageMappingScale
Set pApp = Application
pPage.PrinterChanged pApp.Printer
```

Using map sheets

```
Dim pQF As IQueryFilter
Dim pCursor As IFeatureCursor
Dim pMapSheet As iFeature
Dim pBB As IEnvelope

Set pQF = New QueryFilter
pQF.WhereClause = "system_map_number = '" & SheetName & "'"
Set pCursor = pFClass.Search(pQF, False)
Set pMapSheet = pCursor.NextFeature
Set pBB = pMapSheet.Shape.Envelope

pActiveView.Extent = pBB
```

Legends step 1: create a symbol

```
Dim pSym As ISymbol
Dim pColor As IRgbColor
Dim pSLSym As ISimpleLineSymbol

Set pColor = New RgbColor
pColor.Red = 115
pColor.Green = 223
pColor.Blue = 255
Set pSLSym = New SimpleLineSymbol
pSLSym.color = pColor
pSLSym.Style = esriSLSSolid
pSLSym.Width = 2
Set pSym = pSLSym
```

Legends step 2: create a renderer

```
Dim pRender As IUniqueValueRenderer
Dim iCount As Long
Dim sVal As String, sDesc As String
Dim vSubVals As Variant

Set pRender = New UniqueValueRenderer
With pRender
   .FieldCount = 2
   .Field(0) = "system_name"
   .Field(1) = "maop"
   .FieldDelimiter = ","
   .DefaultSymbol = pSym
End With
pRender.UseDefaultSymbol = False
‘ for example: sVal = “WILLIAMS 1,60”
vSubVals = Split(sVal, ",")
sDesc = vSubVals(0) & " - MAOP " & vSubVals(1) & " psig"
pRender.AddValue sVal, "System Name, MAOP", pSym
pRender.Label(sVal) = sDesc
```

Legends step 3: assign to layer

```
Dim pLayer as ILayer
Dim pTableDef As ITableDefinition

WhereClause = "system_name = '" & sName & _
   "' AND feature_status <> 'Proposed Addition'" & _
   " AND feature_status <> 'Abandoned'“
…
Set pTableDef = pLayer
pTableDef.DefinitionExpression = WhereClause

Dim pGFLayer As IGeoFeatureLayer

Set pGFLayer = pLayer
Set pGFLayer.Renderer = pRender
```

Legends step 4: delete existing legend

```
Dim pDoc As IMxDocument
Dim pLayout As IPageLayout
Dim pGC As IGraphicsContainer
Dim pElement As IElement
Dim pMapSurroundFrame As IMapSurroundFrame

Set pDoc = ThisDocument
Set pLayout = pDoc.PageLayout
Set pGC = pLayout
pGC.Reset
Set pElement = pGC.Next
Do While Not pElement Is Nothing
   If TypeOf pElement Is IMapSurroundFrame Then
      Set pMapSurroundFrame = pElement
      If TypeOf pMapSurroundFrame.MapSurround Is ILegend Then
         pGC.DeleteElement pElement
         Exit Do
      End If
   End If
   Set pElement = pGC.Next
Loop
```

Legends step 5: create new legend

```
Set pActiveView = pMap
pActiveView.ContentsChanged
pDoc.UpdateContents

Dim pEnv As IEnvelope
Dim pID As New UID
Dim pMapFrame As IMapFrame

Set pEnv = New Envelope
pEnv.PutCoords 13.5, 1.25, 16.5, 10.5
pID.Value = "esriCarto.Legend"
Set pMapFrame = pGC.FindFrame(pMap)
Set pMapSurroundFrame = pMapFrame.CreateSurroundFrame(pID, Nothing)
pMapSurroundFrame.MapSurround.Name = "Legend"

Set pElement = pMapSurroundFrame
Set pActiveView = pLayout
pElement.Geometry = pEnv
```

Legends step 6: resize legend

```
Dim pMapSurround As IMapSurround
Dim pNewEnv As IEnvelope

pGC.AddElement pElement, 0
pElement.Activate pActiveView.ScreenDisplay
Set pMapSurround = pLegend
pMapSurround.Refresh
Set pNewEnv = New Envelope
pElement.Geometry = pEnv
pLegend.QueryBounds pActiveView.ScreenDisplay, pEnv, pNewEnv
d = pNewEnv.Height * 3 / pNewEnv.Width
x = pNewEnv.XMin
y = pNewEnv.YMax
If d <= 8 Then y = 9.5 Else y = 10.5
pEnv.PutCoords x, y - d, x + 3, y
pElement.Draw pActiveView.ScreenDisplay, Nothing
pElement.Geometry = pEnv
pMapSurround.FitToBounds pActiveView.ScreenDisplay, pEnv, True
pMapSurround.Refresh
pActiveView.Refresh
```

Tagging text elements

```
Dim pElement As IElement
Dim pText As ITextElement
Dim sText As String, sReturn As String
Dim pProp As IElementProperties
…
Set pText = pElement
Set pProp = pText
sText = pProp.Name
If sText = "" Then
   sText = pText.Text
End If
sReturn = InputBox("Enter name:", "Assign Name", sText)
If sReturn <> "" Then
   pProp.Name = sReturn
End If
```

Populating a tagged element

```
Dim pGC As IGraphicsContainer
Dim pElement As IElement
Dim pText As ITextElement
Dim pProp As IElementProperties

Set pGC = pLayout
pGC.Reset
Set pElement = pGC.Next
Do While Not pElement Is Nothing
  If TypeOf pElement Is ITextElement Then
    Set pText = pElement
    Set pProp = pText
    sText = pProp.Name
    Select Case sText
      Case "date_now"
        pText.text = Format(Now, "mm/dd/yyyy")
    End Select
  End If
  Set pElement = pGC.Next
Loop
```

Export to PDF: get page size

```
DPI = 600
OutFile = "c:\temp\example.pdf"
bEmbedFonts = False

Dim pageUnits As esriUnits
Dim dCF As Double, dWidth As Double, dHeight As Double
Dim pUC As IUnitConverter

Set pDoc = ThisDocument
Set pLayout = pDoc.PageLayout
Set pPage = pLayout.Page
pageUnits = pPage.Units
If pageUnits = esriInches Then
   dCF = 1#
Else
   Set pUC = New UnitConverter
   dCF = pUC.ConvertUnits(1#, esriInches, pageUnits)
End If
pPage.QuerySize dWidth, dHeight
```

Export to PDF: set display transform

```
Dim pEnv As IEnvelope, pOldEnv As IEnvelope, pOldExtent As IEnvelope
Dim devRect As tagRECT, oldRect As tagRECT
Dim pActiveView As IActiveView
Dim pSD As IScreenDisplay
Dim pDT As IDisplayTransformation

Set pEnv = New Envelope
pEnv.PutCoords 0, 0, dWidth, dHeight
devRect.Left = 0
devRect.Right = CInt(dWidth * DPI / dCF)
devRect.Top = 0
devRect.bottom = CInt(dHeight * DPI / dCF)
Set pActiveView = pLayout
Set pOldExtent = pActiveView.Extent
Set pSD = pActiveView.ScreenDisplay
Set pDT = pSD.DisplayTransformation
Set pOldEnv = pDT.Bounds
oldRect = pDT.DeviceFrame
dOldRes = pDT.Resolution
dOldScale = pDT.ScaleRatio
pDT.Bounds = pEnv
pDT.DeviceFrame = devRect
pDT.Resolution = DPI
pDT.ScaleRatio = 1#
```

Export to PDF: do export

```
Dim pExport As IExport
Dim pPixelBoundsEnv As IEnvelope
Dim pExportOpt As IExportVectorOptions
Dim pExportPDF As IExportPDF
Dim hdc As OLE_HANDLE

Set pExport = New ExportPDF
pExport.ExportFileName = OutFile
pExport.Resolution = DPI
Set pPixelBoundsEnv = New Envelope
pPixelBoundsEnv.PutCoords devRect.Left, devRect.Top, devRect.Right, _
   devRect.bottom
pExport.PixelBounds = pPixelBoundsEnv
Set pExportOpt = pExport
pExportOpt.PolygonizeMarkers = True
Set pExportPDF = pExport
pExportPDF.EmbedFonts = bEmbedFonts
pExportPDF.Compressed = True
hdc = pExport.StartExporting
pActiveView.Output hdc, DPI, devRect, Nothing, Nothing
pExport.FinishExporting

‘ (Restore saved settings and refresh)
```

Demo and Questions

Download at:

http://www.pierssen.com/arcgis/misc.htm