APPENDIX D

LISTINGS OF AVENUE WRAPS

Presented in this appendix are the headings for each of the Avenue Wraps™ that were discussed in the previous chapters. To eliminate the retyping of any code, the attached CD includes an ArcMap™ document file (avwraps.mxd), that contains all of the Avenue Wraps™ along with samples which demonstrate the use of the Avenue Wraps™. In fact, the ArcMap document file can be used as a starting point in creating an application. That is, the developer can immediately begin to add new VBA macros and tools to the document file.

Although there are more than 2,000 Avenue requests, 90% of all Avenue programming will probably use only 10% of the Avenue requests. This book presents more than 260 Avenue Wraps, which is approximately 10% of the total number of Avenue requests. As such, it is the authors feelings that we have captured the most prevalent requests, which an Avenue programmer would use. Those which we did not include could probably be written using the techniques that are shown in this Appendix. That is one of the reasons for including this Appendix along with the CD, is that, the reader can review the Avenue Wraps™, see how they were written and perhaps apply this approach in converting other requests that were not included in this book.

One of the issues that became apparent at the outset of our working with ArcObjects™ is that the vocabulary is different from that of Avenue. This, as can be expected, made converting Avenue code into ArcObjects™ somewhat challenging. For example, legends in Avenue are referred to as renderers in ArcObjects™, while themes are now called layers, and so forth. It is hoped that by providing the Avenue programmer with these Avenue Wraps™ we can decrease the ArcObjects™ learning curve, that is, by reviewing the source listings the developer should be able to pick up the ArcObjects™ vocabulary. For example, the reader can find an Avenue request, review the source listing and find the corresponding ArcObjects™ terminology.

Following this introduction, there is a section containing various notes, which describe key points concerning the development of the Avenue Wraps™. It is recommended that these notes be reviewed prior to examining the source listings. In so doing, the methodology presented in the source listings should be a little easier to understand.

As a final note, every programmer has their own style of writing code. In reviewing the listings, the reader may feel that a different approach should have been taken in writing some of the wraps. This very well could be true. As the saying goes, "there is more than one way to skin a cat". What the authors have tried to do in developing the Avenue Wraps™, is not to develop the least amount of code, which more than often is confusing, but rather to present wraparounds that are functional, straightforward and well documented.

Notes regarding the Listings

◗ ArcMap™ employs a single-document interface, while ArcView® employs a multi-document interface, which changes the meaning of the GetActiveDoc request greatly.

◗ In Avenue, the variable *theView* was widely used. This variable has been replaced by the variable *pmxDoc* within the Avenue Wraps™.

◗ Rather than passing objects for layers or tables in the argument list for subroutines or functions, the name of the layer or table is passed as a variable of Variant type.

◗ When setting a field value for either a feature or table record, use the ***Store*** method to write the value to disk, see the Avenue Wraps™ **avSetValue** and **avSetValueG**.

◗ In Avenue, the request ReturnValue was applied to a FTab or VTab in order to extract a value out of a table. In ArcObjects™, the ***Value*** property is used to extract the value from an IFeature or IRow object. The following example should explain:

With Avenue
```
      aFTab = aTheme.GetFTab
      colS = aFTab.FindField("shape")
      pFeature = aFTab.ReturnValue(colS, rec)
      colA = aFTab.FindField("area")
      theArea = aFTab.ReturnValue(colA, rec)
```
With Avenue Wraps
```
      Dim pmxDoc As IMxDocument
      Dim aTheme As Variant
      Dim aFTab As IFields
      Dim aFeatClass As IFeatureClass
      Dim aLayer As IFeatureLayer
      Dim rec, colA As Long
      Dim pFeature As IFeature
      Dim theArea As Double
      Call avGetFTab(pmxDoc, aTheme, _
                     aFTab, aFeatClass, aLayer)
      Call avGetFeature(pmxDoc, aTheme, rec, pFeature)
      colA = aFTab.FindField("area")
      theArea = pFeature.Value(colA)
```

◗ The Avenue requests ChoiceAsString and ListAsString should be substituted with the Avenue Wraps™ **avMsgBoxChoice** and **avMsgBoxList**, respectively.

◗ An example of converted Avenue code, which cycles through a selected set of features to compute a total value, is shown below:

With Avenue

```
theFTab = theTheme.GetFTab
theSel = theFTab.GetSelection
theField = theFTab.FindField("Deposits")
total = 0.0
for each rec in theSel
    deposit = theFTab.ReturnValue(theField, rec)
    total = total + deposit
end
```

With Avenue Wraps

```
Dim pmxDoc As IMxDocument
Dim theTheme As Variant
Dim theFTab As IFields
Dim aFeatClass As IFeatureClass
Dim aLayer As IFeatureLayer
Dim theSel As ISelectionSet
Dim theField, iRec, rec, colA As Long
Dim total, deposit As Double
Dim theSelList As New Collection
Dim pFeat As IFeature
Call avGetFTab(pmxDoc, theTheme, _
               theFTab, aFeatClass, aLayer)
Call avGetSelection(pmxDoc, theTheme, theSel)
theField = theFTab.FindField("Deposits")
total = 0.0
Call avGetSelectionIDs(theSel, theSelList)
For iRec = 1 to theSelList.Count
    rec = theSelList.Item(iRec)
    Set pFeat = aFeatClass.GetFeature(rec)
    deposit = pFeat.Value(theField)
    total = total + deposit
Next
```

◗ In Avenue the @ character could be used to create a point. Use the Avenue Wrap™, **avPointMake** to create the point. For example:

With Avenue

```
aPoint = 5000.0 @ 5000.0
```

With Avenue Wraps

```
Dim aPoint As IPoint
Set aPoint = avPointMake(5000.0, 5000.0)
```

- In Avenue, the request YesNoCancel was applied to the MsgBox Class to determine a course of action from the user. The value that was passed back by the request was a Boolean, using the Avenue Wraps, the returned value is an integer which will be equal to one of the predefined VB constants. The following example should explain:

With Avenue

```
      ians = MsgBox.YesNoCancel(Msg, Heading, Default)
      if (ians = Nil) then
         .... do something
      end
      if (ians.Not) then
         .... do something
      end
```

With Avenue Wraps

```
      Dim Msg, Heading As Variant
      Dim Default As Boolean
      Dim ians As Integer
      Call avMsgBoxYesNoCancel(Msg, Heading, Default, _
                                   ians)
      If (ians = vbCancel) then
         .... do something
      End If
      If (ians = vbNo) then
         .... do something
      End If
      If (ians = vbYes) then
         .... do something
      End If
```

- The Avenue request IsNoDataClassDisplayed should be substituted with the ArcObjects™ property, ***UseDefaultSymbol***.

With Avenue

```
      noData = aLegend.IsNoDataClassDisplayed
```

With Avenue Wraps

```
      Dim pUniqueRend As IUniqueValueRenderer
      Dim noData As Boolean
      noData = pUniqueRend.UseDefaultSymbol
```

- The Avenue request ClearMsg can be replaced with the Avenue Wrap **avClearStatus**. The following example should explain:

With Avenue

```
      av.ClearMsg
```

With Avenue Wraps

```
      Call avClearStatus
```

- In Avenue, the programmer could have an if ... then ... end statement on a single data line. In VB or VBA, if an if ... then ... end statement appears in red (denoting an error condition) the data line can be decomposed into a multi-line statement. The following example should explain:

In Avenue

```
      if (a = 2.0) then b = 4.0 end
```

In VB/VBA

```
      Dim a, b As Double
      If (a = 2.0) then
         b = 4.0
      End If
```

- The object id (OID) values for shapefiles start at 0 and increase sequentially by one, while for personal geodatabases they start at 1.

- The Avenue request ReturnFamilies could be used to get a list of the available Windows fonts. The Avenue Wrap **avGetWinFonts** can be substitued in its place. The following example should explain:

With Avenue

```
      aFontManager = FontManager.The
      aFontList = aFontManager.ReturnFamilies
```

With Avenue Wraps

```
      Dim aFontList As New Collection
      Call avGetWinFonts(aFontList)
```

- The Avenue request FindAllByClass could be used to find all of the graphic text elements in a view. Using Avenue Wraps the following procedure could be used to accomplish the same task:

With Avenue

```
      graphList = theView.GetGraphics
      gTextList = graphList.FindAllByClass(GraphicText)
```

With Avenue Wraps

```
      Dim graphList As New Collection
      Dim i As Long
      Dim pElement As IElement
      Dim gTextList As New Collection
      Call avViewGetGraphics(graphList)
      If (graphList.Count > 0) Then
         For i = 1 To graphList.Count
             Set pElement = graphList.Item(i)
             If TypeOf pElement Is ITextElement Then
                gTextList.Add pElement
             End If
         Next
      End If
```

◗ The Avenue request ReturnProjected could be used to project a feature into a specific projection. The ArcObjects™ method, ***Project*** can be applied to an IGeometry object to accomplish the same task. The following example shows how a feature can be projected into the map's (pMap) current projection (pSpatialReference). Note that it is the geometry (aShape) of the feature that is actually processed.

With Avenue

```
      newShape = aShape.ReturnProjected(thePrj)
```

With Avenue Wraps

```
      Dim pmxDoc As IMxDocument
      Dim pMap As IMap
      Dim pSpatialReference As ISpatialReference
      Dim aShape As IGeometry
      Set pmxDoc = Application.Document
      Set pMap = pmxDoc.FocusMap
      Set pSpatialReference = pMap.SpatialReference
      aShape.Project pSpatialReference
```

◗ The Avenue request ReturnUnProjected could be used to unproject a feature into its own natural projection. The ArcObjects™ method, ***Project*** can be applied to an IGeometry object to accomplish the same task. The following example shows how a feature (aFeature) can be unprojected from the map's (pMap) current projection (pSpatialReference) into the projection of the layer (pFeatureClass) in which the feature resides. Note that it is the geometry (aShape) of the feature that is actually processed.

With Avenue

```
      theNewShape = aShape.ReturnUnProjected(thePrj)
```

With Avenue Wraps

```
      Dim pmxDoc As IMxDocument
      Dim pMap As IMap
      Dim pSpatialReference As ISpatialReference
      Dim pObjectClass As IObjectClass
      Dim aFeature As IFeature
      Dim pFeatureClass As IFeatureClass
      Dim pGeoDataSet As IGeoDataset
      Dim aShape As IGeometry
      Set pmxDoc = Application.Document
      Set pMap = pmxDoc.FocusMap
      Set pSpatialReference = pMap.SpatialReference
      Set pObjectClass = aFeature.Class
      Set pFeatureClass = pObjectClass
      Set pGeoDataSet = pFeatureClass
      Set aShape.SpatialReference = pSpatialReference
      aShape.Project pGeoDataSet.SpatialReference
```

- The Avenue request SetAlias could be used to assign an alias to a field. The ArcObjects™ property, ***AliasName*** can be applied to an IFieldEdit object to accomplish the same task. The following example shows how a field object can be assigned an alias. This example processes a FTab but will work for VTab's as well.

  With Avenue
  ```
        col = aFTab.FindField("aField")
        col.SetAlias("New_Field_Name")
  ```
  With Avenue Wraps
  ```
        Dim aFTab As IFields
        Dim col As Long
        Dim pField As IField
        Dim pFieldEdit As IFieldEdit
        col = aFTab.FindField("aField")
        Set pField = aFTab.Field(col)
        Set pFieldEdit = pField
        pFieldEdit.AliasName("New_Field_Name")
  ```

- The Avenue request GetAlias could be used to get the alias that is assigned to a field, using the Avenue Wraps™ **avGetAlias** wraparound, Avenue code using GetAlias would be converted as follows:

  With Avenue
  ```
        col = aFTab.FindField("aField")
        anAlias = col.GetAlias
  ```
  With Avenue Wraps
  ```
        Dim aFTab As IFields
        Dim col As Long
        Dim anAlias As String
        col = aFTab.FindField("aField")
        anAlias = avGetAlias(col)
  ```

- **ArcGIS 9.x** users should disregard any references to the esriCore library. At 9.x the esriCore library was replaced with numerous other libraries. Note that it is not necessary to include the name of the library in the declaration statement, such as IField. Solely for clarification is the name of the library included in the declaration statement.

- The remainder of this Appendix is in a .pdf file called AppendixD.pdf, which is stored in the **CEDRA Avenue Wraps** distribution directory. The Adobe Acrobat Reader software can be used to print out this file, as well as, any other software that can process a .pdf file.

How to Install the CEDRA Avenue Wraps Document File:

Step 1:

The **CEDRA Avenue Wraps**™ software requires approximately five (5) megabytes of disk space and will operate under **Windows NT**®**, Windows 2000**® **and Windows XP**®. One top level directory, whose default name is **CEDRA**, will be created. Should the user wish to use a different name, the user can do so. Within the **CEDRA** directory a single **ArcMap**™ document file called **avwraps.mxd** will be stored. This file contains all of the modules and forms that comprise the **Avenue Wraps**.

It is assumed in this installation discussion, that the "**C:**" partition will be used to contain the software. If this is not the case, it is possible to substitute the appropriate drive identifier when performing the installation. It is also assumed that the individual installing the software is somewhat familiar with **PC** terminology, has a working knowledge of the **PC** and the available text editors that are installed on the **PC**, and is somewhat familiar with **ArcGIS**™.

Note that in order to operate **CEDRA Avenue Wraps**, **ArcGIS**™ Version 8.1.2 or higher needs to be installed on the **PC**. The user should verify that this requirement is satisfied this time.

The **CEDRA Avenue Wraps** software consists of a single CD and contains the **ArcMap** document file in a compressed file format. Prior to installing the software, a partition on the **PC** should be found that contains the necessary amount of free disk space, five (5) megabytes.

Step 2:

The contents of the **CEDRA Avenue Wraps** software can now be extracted and stored onto the **PC**. The CD should now be inserted into the appropriate drive.

Step 3:

Select the **Start** button from the task bar followed by selecting the **Run...** menu item.

Step 4:

The **CEDRA** software installation program can be invoked by typing:

```
D:SETUP
```

If the CD drive identifier is something other than D, the user should make the appropriate substitution in the above command.

The program will then pose a series of screens guiding the user through the installation process. Once the final screen has been displayed, the program will decompress the document file, and store the file in the appropriate directory location. After the file has been decompressed, the user can invoke **ArcMap** and open the document file.

How to Invoke the CEDRA Avenue Wraps Document File:

This version of **CEDRA Avenue Wraps**™ is available as an **ArcMap**™ document file called **avwraps.mxd**. This document file contains only the modules and forms that comprise the **Avenue Wraps**. Using **avwraps.mxd** the developer can create additional forms, modules and controls to establish a custom application. To begin using the **CEDRA Avenue Wraps** perform the following:

➤1 Invoke the **ArcMap**™ program.

➤2 **Click** in the radial button to the left of the *An existing map:* label, then **click** the **OK** button.

➤3 **Navigate** to the **\cedra** directory, **click** on the **avwraps.mxd** file name, and then **click** the **OK** button.

➤4 At this point the developer can begin reviewing the **CEDRA Avenue Wraps** and building a custom application.

Sample Data:

In addition to the **ArcMap** document file, **avwraps.mxd**, seven VBA modules and two shapefiles are included containing sample code and data, which illustrate the use of the **CEDRA Avenue Wraps**. These samples include:

Module1.bas	Sample illustrating how to process graphics and symbols, the graphics that are created are based upon an arbitrary coordinate system.
Module2.bas	Sample illustrating how to process feature geometry. This is done by using the first selected feature in a polygon theme.
Module3.bas	Sample illustrating how to create a Shapefile and add a feature to it. The sample will also show how an operation can be defined.
Module4.bas	Sample illustrating how to perform various shape editing operations. This sample requires seven polygon features and one polyline feature be selected prior to executing this macro. The first selected polygon and the selected polyline features will be used in a split operation. The remaining selected polygons will be used to demonstrate (a) merging, (b) intersecting and (c) unioning operations. The shapefiles **L_0pg** and **L_0pl**, which are included in the distribution set, can be added to **ArcMap** and used in this sample.
Module5.bas	Sample illustrating how to create, add records, populate and summarize a table.
Module6.bas	Sample illustrating how to create a new shapefile that has a default spatial reference and three attributes using a name that the user enters in a file dialog box. The shapefile is to contain Polyline features and will be added to the map once it has been created.
Module7.bas	Sample illustrating how to create various types of message boxes.

To import and execute a sample perform the following:

➤**1** Invoke the **ArcMap**™ program.

➤**2** **Click** in the radial button to the left of the *An existing map:* label, then **click** the **OK** button.

➤3 **Navigate** to the **\cedra** directory, **click** on the **avwraps.mxd** file name, and then **click** the **OK** button.

➤4 Click at the **Tools** menu and then at the **Macros** and **Visual Basic Editor** sub-menus.

➤5 Click at the plus sign, **+**, to the left of the **Project (avwraps.mxd)** label in the project window to expand the project document.

➤6 Click at the **File** menu and then at the **Import File...** sub-menu.

➤7 **Navigate** to the **\cedra** directory, **click** on the desired sample module file name, and then **click** the **Open** button.

➤8 Click at the plus sign, **+**, to the left of the **Modules** label in the project window to expand the module document.

➤9 Scroll down the list of modules and find the sample module file that was imported. Once found, double-click on the name of the module to open the module.

➤10 Click at the Run Sub/UserForm tool (▶) to execute the sample code.

```
Public  Sub  avaClassMake(aclass,  shapeList,  theFeat)
' *                                                                    *
' *   PURPOSE:   TO CREATE A SPECIAL FEATURE OBJECT FROM A POINT LIST    *
' *                                                                    *
' *  GIVEN:     aClass     = the type of special feature               *
' *                          11 for PolyLineM                          *
' *                          12 for PolyLineZ                          *
' *                          13 for PolygonM                           *
' *                          14 for PolygonZ                           *
' *                          15 for PointM                             *
' *                          16 for PointZ                             *
' *                          17 for MultiPointM                        *
' *                          18 for MultiPointZ                        *
' *                          31 for PolyLineM and PolyLineZ            *
' *                          32 for PolygonM and PolygonZ              *
' *                          33 for PointM and PointZ                  *
' *                          34 for MultiPointM and MultiPointZ        *
' *                          41 for PolyLine                           *
' *                          42 for Polygon                            *
' *                          43 for Point                              *
' *                          44 for MultiPoint                         *
' *             shapeList = the list of points comprising the feature  *
' *                          structure of shapeList is:                *
' *                          Item 1: number of parts                   *
' *                          Item 2: number of points in part 1        *
' *                          Item 3: x value of point 1 in part 1      *
' *                          Item 4: y value of point 1 in part 1      *
' *                          Item 5: z value of point 1 in part 1      *
' *                          Item 6: m value of point 1 in part 1      *
' *                          Item 7: id value of point 1 in part 1     *
' *                          Item 8: Repeat Items 3 - 7 for each       *
' *                                  point                             *
' *                          Repeat Items 2 - 8 for each part          *
' *                                                                    *
' *  RETURN:    theFeat    = the special feature                       *
' *                                                                    *
' *  Dim aClass As Integer, shapeList As New Collection                *
' *  Dim theFeat As IPoint, IMultiPoint, IPolyline, or IPolygon        *
' *                                                                    *
Public  Function  avAddDoc(aDoc  As  IUnknown)
' *                                                                    *
' *  PURPOSE:  TO ADD A LAYER OR TABLE TO THE MAP                      *
' *                                                                    *
' *  GIVEN:     aDoc      = the document to be added                   *
' *                                                                    *
' *  RETURN:    avAddDoc = error flag (0 = no error, 1 = error)        *
' *                                                                    *
' *  NOTE:       When adding multiple layers and/or tables to the map  *
' *              in succession, it may be appropriate to set the global *
' *              variable ugUpdateTOC to False prior to adding the     *
' *              first layer or table. Then, prior to adding the last  *
' *              layer or table set ugUpdateTOC to True. In so doing,  *
' *              the TOC will be refreshed only once and not every time *
' *              a layer or table is added (When using the DLL version *
' *              of Avenue Wraps set the property avwraps.ugUpdateTOC  *
' *              to True or False, do not use the global ugUpdateTOC)  *
' *                                                                    *
' *  Dim aDoc As IUnknown                                              *
' *  Dim avAddDoc As Integer                                           *
' *                                                                    *
Public  Function  avAddFields(pmxDoc  As  IMxDocument,  theTheme,  theFieldS)
' *                                                                    *
' *  PURPOSE:  TO ADD FIELDS INTO A LAYER OR TABLE                     *
' *                                                                    *
' *  GIVEN:     pmxDoc       = the active view                         *
' *               theTheme     = the theme or table to be processed    *
' *               theFields    = list of fields to be added, the items in *
' *                              this list are IFieldEdit objects, not  *
' *                              strings                               *
' *                                                                        *
' *   RETURN:    avAddFields = error flag (0 = no error, 1 = error)        *
' *                                                                        *
' *   NOTE:      In order to add fields into a layer or table the          *
' *                editor can not be in an edit state, this routine will   *
' *                stop the editor if the editor is in an edit state       *
' *                thereby saving any changes that may have been made,     *
' *              prior to adding the fields                                *
' *                                                                        *
' *   Dim pmxDoc As IMxDocument                                            *
' *   Dim theTheme As Variant, theFields As New Collection                 *
' *   Dim avAddFields As Integer                                           *
' *                                                                        *
```
Public Function avAddRecord(pmxDoc As IMxDocument, theTheme) As Long
```
' *                                                                        *
' *   PURPOSE:  TO ADD A RECORD INTO A LAYER OR TABLE                      *
' *                                                                        *
' *   GIVEN:     pmxDoc       = the active view                            *
' *                theTheme     = the theme or table to be processed       *
' *                                                                        *
' *   RETURN:    avAddRecord  = the id of the record that was added, if    *
' *                              a record can not be added will be -1      *
' *                                                                        *
' *   Dim pmxDoc As IMxDocument                                            *
' *   Dim theTheme As Variant                                              *
' *   Dim avAddRecord As Long                                              *
' *                                                                        *
```
Public Sub avAsList(theFeature As IFeature, shapeList)
```
' *                                                                        *
' *   PURPOSE:   TO CREATE A LIST CONTAINING THE POINTS THAT COMPRISE      *
' *              A POINT, LINE OR POLYGON FEATURE                          *
' *                                                                        *
' *   GIVEN:     theFeature = feature to be processed                      *
' *                                                                        *
' *   RETURN:    shapeList   = the shape's list of points for each part    *
' *                              structure of shapeList is:                *
' *                               Item 1: collection of points in part 1   *
' *                              Repeat Item 1 for each part               *
' *                              So that, shapeList is a list of           *
' *                               collections with each collection         *
' *                             containing points                          *
' *                                                                        *
' *   NOTE:        Use subroutine avPlAsList when an IGeometry object is   *
' *              known and not an IFeature object                          *
' *                                                                        *
' *   Dim theFeature As IFeature                                           *
' *   Dim shapeList As New Collection                                      *
' *                                                                        *
```
Public Sub avAsList2(theFeature As IFeature, shapeList)
```
' *                                                                        *
' *   PURPOSE:   TO CREATE A LIST CONTAINING THE POINTS THAT COMPRISE      *
' *              A LINE OR POLYGON FEATURE                                 *
' *                                                                        *
' *   GIVEN:     theFeature = feature to be processed                      *
' *                                                                        *
' *   RETURN:    shapeList   = the list of points comprising the feature   *
' *                              structure of shapeList is:                *
' *                              Item 1: number of parts                   *
' *                               Item 2: number of points in part 1       *
' *                              Item 3: point 1 in part 1                 *
' *                              Item 4: point 2 in part 1                 *
' *                             Item 5:         .                          *
' *                             Item 6:         .                          *
' *                              Item n: point n in part 1                 *
' *                               Repeat Items 2 - n for each part         *
' *                                                                        *
' *   NOTE:        Use subroutine avAsList3 when an IGeometry object is    *
' *              known and not an IFeature object                          *
' *                                                                        *
' *  Dim theFeature As IFeature                                             *
' *  Dim shapeList As New Collection                                        *
' *                                                                         *
Public Sub avAsList3(theShape As IGeometry, shapeList)
' *                                                                         *
' *  PURPOSE:  TO CREATE A LIST CONTAINING THE POINTS THAT COMPRISE         *
' *            A LINE OR POLYGON                                            *
' *                                                                         *
' *  GIVEN:    theShape  = feature to be processed                          *
' *                                                                         *
' *  RETURN:   shapeList = the list of points comprising the feature        *
' *                        structure of shapeList is:                       *
' *                        Item 1: number of parts                          *
' *                        Item 2: number of points in part 1               *
' *                        Item 3: point 1 in part 1                        *
' *                        Item 4: point 2 in part 1                        *
' *                        Item 5:         .                                *
' *                        Item 6:         .                                *
' *                        Item n: point n in part 1                        *
' *                        Repeat Items 2 - n for each part                 *
' *                                                                         *
' *  NOTE:     Use subroutine avAsList2 when an IFeature object is          *
' *            known and not an IGeometry object                            *
' *                                                                         *
' *  Dim theShape As IGeometry                                              *
' *  Dim shapeList As New Collection                                        *
' *                                                                         *
Public Function avAsPolygon(pInput As IUnknown) As IGeometry
' *                                                                         *
' *  PURPOSE:  CONVERT INPUT INTO POLYGON GEOMETRY                          *
' *                                                                         *
' *  GIVEN:    pInput      = the input to be converted                      *
' *                                                                         *
' *  RETURN:   avAsPolygon = polygon geometry                               *
' *                                                                         *
' *  Dim pInput As IUnknown                                                 *
' *  Dim avAsPolygon As IGeometry                                           *
' *                                                                         *
Public Sub avAsTokens(theString, delString, UpperLower, theList, nWords)
' *                                                                         *
' *  PURPOSE:  Read a text string, a word delineator and an indicator       *
' *              whether to change all characters to upper or lower         *
' *            case characters, and                                         *
' *              1. Remove the leading and trailing blank spaces; and       *
' *              2. Create a list of words from the contents of the         *
' *                 read text string excluding therefrom any blank          *
' *                 spaces that may be present between the words; and       *
' *              3. Compute the number of words in the returned list.       *
' *                                                                         *
' *  GIVEN:    theString  = the input string                                *
' *            delString  = the word delineator                             *
' *              UpperLower = U - change all characters to upper case       *
' *                         = L - change all characters to lower case       *
' *                         = X - no change in terms of case and do not     *
' *                               trim leading/trailing characters          *
' *                         = any other character - no change               *
' *                                                                         *
' *  RETURN:   theList    = the returned list of words                      *
' *            nWords     = number of words extracted                       *
' *                                                                         *
' *  NOTE:     When the word delineator is a single blank character         *
' *              and the first character in any word that is extracted      *
' *              is the TAB character, the TAB character is removed         *
' *              from the word as long as UpperLower is not equal to X      *
' *                                                                         *
' *  Dim theString As String, delString As String                           *
' *  Dim UpperLower As String                                               *
' *  Dim theList As New Collection, nWords As Integer                       *
' *                                                                         *
```

```
Public  Function  avBasicTrim(theString,  LeadChar,  TrailChar)
' *                                                                   *
' *   PURPOSE:   REMOVE FROM A GIVEN STRING THE SPECIFIED LEADING     *
' *              AND/OR TRAILING CHARACTERS                           *
' *                                                                   *
' *   GIVEN:     theString    = the given string to be processed      *
' *               LeadChar     = the characters to be removed at the  *
' *                              start of the given string            *
' *               TrailChar    = the characters to be removed at the  *
' *                              end of the given string              *
' *                                                                   *
' *   RETURN:    avBasicTrim = the resultant string                   *
' *                                                                   *
' *   NOTE:       Blank characters are not removed from theString     *
' *                                                                   *
' *   Dim theString As String, LeadChar As String, TrailChar As String *
' *   Dim avBasicTrim As String                                       *
' *                                                                   *
Public  Sub  avBitmapClear(psTableSel  As  ISelectionSet,  theRcrd)
' *                                                                   *
' *   PURPOSE:   REMOVE A RECORD FROM THE SELECTED SET FOR A LAYER OR *
' *              TABLE                                                *
' *                                                                   *
' *   GIVEN:     psTableSel = selection set for a theme or table      *
' *               theRcrd     = record to be removed from the selection *
' *                                                                   *
' *   RETURN:    nothing                                              *
' *                                                                   *
' *   Dim psTableSel As ISelectionSet                                 *
' *   Dim theRcrd As Long                                             *
' *                                                                   *
Public  Sub  avBitmapClearAll(psTableSel  As  ISelectionSet)
' *                                                                   *
' *   PURPOSE:   REMOVE ALL RECORDS FROM THE SELECTED SET FOR A LAYER *
' *              OR TABLE                                             *
' *                                                                   *
' *   GIVEN:     psTableSel = selection set for a theme or table      *
' *                                                                   *
' *   RETURN:    nothing                                              *
' *                                                                   *
' *   Dim psTableSel As ISelectionSet                                 *
' *                                                                   *
Public  Sub  avBitmapSet(pmxDoc  As  IMxDocument,  theTheme,  theRcrd)
' *                                                                   *
' *   PURPOSE:   ADD A RECORD TO THE SELECTED SET FOR A LAYER OR TABLE *
' *                                                                   *
' *   GIVEN:     pmxDoc    = the active view                          *
' *               theTheme = the theme or table to be processed       *
' *               theRcrd  = the record to be added to the selection  *
' *                                                                   *
' *   RETURN:    nothing                                              *
' *                                                                   *
' *   Dim pmxDoc As IMxDocument                                       *
' *   Dim theTheme As Variant, theRcrd As Long                        *
' *                                                                   *
Public  Function  avCalculate(pmxDoc  As  IMxDocument,  theTheme,  _
                               aCalcString, aField) As Integer
' *                                                                   *
' *   PURPOSE:   TO APPLY A CALCULATION TO A FIELD IN A LAYER OR TABLE *
' *                                                                   *
' *   GIVEN:     pmxDoc       = the active view                       *
' *               theTheme     = name of theme or table to be processed *
' *               aCalcString = calculation string to be applied      *
' *                             sample string field equation          *
' *                             aCalcStr = """abcd"""                 *
' *                             sample numeric field equation         *
' *                             aCalcStr = "([SLN] - " + CStr(ii) + ")" *
' *               aField       = field to be populated (index value)  *
' *                                                                   *
' *   RETURN:    avCalculate = error flag as noted below             *
' *                             0 : no error                         *
' *                              1 : theme or table not found        *
' *                               2 : error in performing calculation *
' *                             3 : no records selected              *
' *                               4 : an edit session has not been started *
' *                                                                  *
' *   NOTE:       (a) If the layer or table contains selected records, *
' *                    then only the selected records will be processed, *
' *                    if there are no selected records, then the entire *
' *                  table will be processed                         *
' *               (b) If the layer or table to be processed supports *
' *                    being editted outside of an edit session then it *
' *                   is not necessary to make the layer or table    *
' *                    editable. If the layer or table does not support *
' *                    this and the editor is not in an edit state this *
' *                    function returns the error code of 4 (see above) *
' *               (c) If the layer or table has any joins, no progress *
' *                    bar will be displayed, otherwise a progress bar *
' *                  will be displayed                               *
' *               (d) When a layer or table has a join note that it is *
' *                    required to prefix the field name with the name of *
' *                    the layer or table (do not use the alias name of *
' *                    the layer or table) when getting the field index *
' *                    value, for example if layer ABCD is joined to the *
' *                    table EFG and the attribute 123 appears in ABCD, *
' *                   the argument in FindField would be:            *
' *                        aField = theFTab.FindField("ABCD.123")    *
' *               (e) The syntax for aCalcString shown above works for *
' *                    both shapefiles and personal geodatabases     *
' *                                                                  *
' *  Dim pmxDoc As IMxDocument                                       *
' *   Dim theTheme As Variant, aCalcString As String, aField As Long *
' *  Dim avCalculate As Integer                                      *
' *                                                                  *
```

Public Sub avCheckEdits(pEditor As IEditor, pDataSet As IDataset)

```
' *                                                                  *
' *   PURPOSE:   TO PERFORM CHECKS ON THE EDITING OF DATA            *
' *                                                                  *
' *   GIVEN:     pEditor  = the ArcMap Editor extension              *
' *                pDataSet = the dataset to be processed, if NOTHING is *
' *                           specified and if the editor is in an edit *
' *                           state, the editor is stopped saving any *
' *                          edits that may have been made           *
' *                                                                  *
' *  RETURN:    nothing                                              *
' *                                                                  *
' *   NOTE:       This routine first checks if the editor is in an edit *
' *                state, if not, this routine does nothing, if it is in *
' *                an edit state it will check if the dataset passed in *
' *                is currently being edited, if not the routine saves *
' *                the edits on the dataset currently being edited and *
' *                starts the editor on the dataset that is passed in *
' *                                                                  *
' *  Dim pEditor As IEditor                                          *
' *  Dim pDataSet As IDataset                                        *
' *                                                                  *
```

Public Function avCircleMakeXY(xPt, yPt, rad) As ICurve

```
' *                                                                  *
' *   PURPOSE:   TO CREATE A CIRCLE FROM COORDINATES AND A RADIUS    *
' *                                                                  *
' *   GIVEN:     xPt             = x coordinate of circle center     *
' *               yPt             = y coordinate of circle center    *
' *              rad            = radius of circle                   *
' *                                                                  *
' *   RETURN:    avCircleMakeXY = the curve feature                  *
' *                                                                  *
' *   Dim xPt As Double, yPt As Double, rad As Double                *
' *  Dim avCircleMakeXY As ICurve                                    *
' *                                                                   *
Public  Function  avClean(aShape1  As  IGeometry)  As  IGeometry
' *                                                                   *
' *   PURPOSE:   TO VERIFY AND ENFORCE THE CORRECTNESS OF A SHAPE      *
' *                                                                   *
' *  GIVEN:     aShape1 = shape to be cleaned                         *
' *                                                                   *
' *   RETURN:    avClean = new shape reflecting the cleaning          *
' *                                                                   *
' *  Dim aShape1 As IGeometry                                         *
' *  Dim avClean As IGeometry                                         *
' *                                                                   *
Public  Sub  avClearSelection(pmxDoc  As  IMxDocument,  theTheme)
' *                                                                   *
' *   PURPOSE:   CLEAR THE SELECTION SET FOR A LAYER OR TABLE         *
' *                                                                   *
' *  GIVEN:     pmxDoc    = the active view                           *
' *                theTheme = the theme or table to be processed, if NULL *
' *                           all selected features in all themes will be *
' *                         deselected                                *
' *                                                                   *
' *  RETURN:    nothing                                               *
' *                                                                   *
' *  Dim pmxDoc As IMxDocument                                        *
' *  Dim theTheme As Variant                                          *
' *                                                                   *
Public  Sub  avClearSelection2(pmxDoc  As  IMxDocument,  theTheme)
' *                                                                   *
' *   PURPOSE:   CLEAR THE SELECTION SET FOR A LAYER OR TABLE         *
' *                                                                   *
' *  GIVEN:     pmxDoc    = the active view                           *
' *                theTheme = the theme or table to be processed, if NULL *
' *                           is specified all selected features in all   *
' *                         themes will be deselected                  *
' *                                                                   *
' *  RETURN:    nothing                                               *
' *                                                                   *
' *   NOTE:       When a feature layer is being processed, the display  *
' *               is not updated to reflect the deselection of the      *
' *               features. This is useful when performing loops where  *
' *               it is not necessary to have the screen redrawn after  *
' *               each iteration within the loop (since refreshing the  *
' *               screen is slow this subroutine can be very useful)    *
' *                                                                   *
' *  Dim pmxDoc As IMxDocument                                        *
' *  Dim theTheme As Variant                                          *
' *                                                                   *
Public  Sub  avClearStatus()
' *                                                                   *
' *  PURPOSE:  CLEAR THE STATUS BAR AREA                              *
' *                                                                   *
' *  GIVEN:     nothing                                               *
' *                                                                   *
' *  RETURN:    nothing                                               *
' *                                                                   *
Public  Function  avClone(theObject  As  IUnknown)  As  IClone
' *                                                                   *
' *   PURPOSE:   MAKE A NEW OBJECT BY COPYING AN EXISTING OBJECT      *
' *                                                                   *
' *   GIVEN:     theObject = object which is to be copied             *
' *                                                                   *
' *  RETURN:    avClone    = copy of the object                       *
' *                                                                   *
' *  Dim theObject As IUnknown                                        *
' *  Dim avClone As IClone                                            *
' *                                                                   *
Public  Sub  avConvertArea(theValue,  fromUnit,  toUnit)
' *                                                                   *
' *   PURPOSE:   CONVERT AN AREA VALUE FROM ONE UNIT INTO ANOTHER     *
' *                                                                     *
' *  GIVEN:     theValue = the value to be converted                    *
' *              fromUnit = the from unit of measure                    *
' *                          1 : Inches            3 : Feet             *
' *                          4 : Yards             5 : Miles            *
' *                           7 : Millimeters       8 : Centimeters     *
' *                          9 : Meters           10 : Kilometers       *
' *              toUnit   = the to unit of measure                      *
' *                          3 : Feet               9 : Meters          *
' *                                                                     *
' *  RETURN:   nothing                                                  *
' *                                                                     *
' *  NOTE:      The argument theValue is modified by this procedure     *
' *                                                                     *
' *  Dim theValue As Variant                                            *
' *  Dim fromUnit As esriUnits, toUnit As esriUnits                     *
' *                                                                     *
Public Sub avConvertDistance(theValue, fromUnit, toUnit)
' *                                                                     *
' *  PURPOSE:   CONVERT A DISTANCE VALUE FROM ONE UNIT INTO ANOTHER     *
' *                                                                     *
' *  GIVEN:     theValue = the value to be converted                    *
' *              fromUnit = the from unit of measure                    *
' *                          1 : Inches            3 : Feet             *
' *                          4 : Yards             5 : Miles            *
' *                           7 : Millimeters       8 : Centimeters     *
' *                          9 : Meters           10 : Kilometers       *
' *                         12 : Decimeters                             *
' *              toUnit   = the to unit of measure                      *
' *                          1 : Inches            3 : Feet             *
' *                           7 : Millimeters       8 : Centimeters     *
' *                          9 : Meters                                 *
' *                                                                     *
' *  RETURN:   nothing                                                  *
' *                                                                     *
' *  NOTE:      The argument theValue is modified by this procedure     *
' *                                                                     *
' *  Dim theValue As Variant                                            *
' *  Dim fromUnit As esriUnits, toUnit As esriUnits                     *
' *                                                                     *
Public Sub avCreateTable(pTable As iTable, pCursor As ICursor, filName)
' *                                                                     *
' *  PURPOSE:   TO CREATE A NEW dBASE FILE FROM A TABLE USING DATA IN   *
' *              A CURSOR AND ADD IT TO THE DOCUMENT                    *
' *                                                                     *
' *  GIVEN:     pTable  = ITable object to be processed                 *
' *                 pCursor = ICursor object containing the data that will *
' *                          be written to the new .dbf file            *
' *                filName = name of the new dBase file to be created,  *
' *                          if the name does not contain a complete    *
' *                          pathname the current working directory     *
' *                          will be used, some examples include:       *
' *                                 c:\project\test\atable.dbf          *
' *                                atable.dbf                           *
' *                                                                     *
' *  RETURN:   nothing                                                  *
' *                                                                     *
' *  NOTE:      (a) If the new dBase file that is to be created exists  *
' *                  on disk, it will be deleted and then rewritten     *
' *             (b) If the new dBase table that is to be created        *
' *                  exists in the document, it will be removed prior   *
' *                 to adding it back in                                *
' *              (c) The argument filName can or can not contain the    *
' *                 .dbf extension                                      *
' *                                                                     *
' *  Dim pTable As ITable, pCursor As ICursor, filName As String        *
' *                                                                     *
```

```
Public  Function  avDeleteDS(name  As  String)
' *                                                                    *
' *   PURPOSE:   DELETE A DATASET SUCH AS A SHAPEFILE OR DBASE FILE     *
' *                                                                    *
' *   GIVEN:     name         = name of the dataset to be deleted, if the *
' *                               name does not contain a complete pathname *
' *                               the current working directory will be   *
' *                              used, some examples of name for a        *
' *                             shapefile:                                *
' *                                 c:\project\test\l_0ln                 *
' *                             access database:                          *
' *                                 c:\project\test\montgomery            *
' *                             dBase file:                               *
' *                                 c:\project\test\table                 *
' *                               dataset within an access database:      *
' *                                  G_Grid c:\project\test\L_0.mdb       *
' *                                                                    *
' *   RETURN:    avDeleteDS = error flag (0 = no error, 1 = error)      *
' *                                                                    *
' *   NOTE:      (a) The dataset must not appear in the Table of        *
' *                  Contents if it does an error will be generated,    *
' *                  use the subroutine avRemoveDoc to remove the       *
' *                  dataset from the Table of Contents before calling  *
' *                this function                                        *
' *              (b) If the name passed in contains an extension such   *
' *                  as .shp, .mdb, .dbf, etc., it will be stripped off *
' *                 and no error will be generated                      *
' *              (c) When a dataset within an access database is to be  *
' *                  deleted, the programmer specifies the name of the  *
' *                  dataset and the personal geodatabase with at least *
' *                  one blank character (space) separating the two     *
' *                  items. In this mode the full pathname for the      *
' *                  personal geodatabase must be specified.            *
' *                                                                    *
' *   Dim name As String                                               *
' *   Dim avDeleteDS As Integer                                        *
' *                                                                    *
Public  Function  avDirExists(name)  As  Boolean
' *                                                                    *
' *   PURPOSE:   DETERMINE IF A DIRECTORY EXISTS OR NOT                 *
' *                                                                    *
' *   GIVEN:     name          = name of directory to be checked, if the  *
' *                               directory is not in the current folder a *
' *                               complete pathname must be specified     *
' *                                                                    *
' *   RETURN:    avDirExists = existence flag (true = yes, false = no)  *
' *                                                                    *
' *   Dim name As String                                               *
' *   Dim avDirExists As Boolean                                       *
' *                                                                    *
Public  Sub  avDisplayInvalidate(aFlag)
' *                                                                    *
' *   PURPOSE:   TO REFRESH OR REDRAW THE ACTIVE DISPLAY                *
' *                                                                    *
' *   GIVEN:     aFlag    = when to redraw (true = at the next refresh, *
' *                          false = immediately)                       *
' *                                                                    *
' *   RETURN:    nothing                                               *
' *                                                                    *
' *   Dim aFlag As Boolean                                             *
' *                                                                    *
Public  Sub  avDocActivate(aName)
' *                                                                    *
' *   PURPOSE:   TO ACTIVATE A NEW VIEW (DATA FRAME)                    *
' *                                                                    *
' *   GIVEN:     aName    = name of the document to be activated        *
' *                                                                    *
' *   RETURN:    nothing                                               *
' *                                                                    *
' *  Dim aName As String                                              *
' *                                                                   *
Public  Sub  avExecute(aCommand)
' *                                                                   *
' *  PURPOSE:  TO EXECUTE A SYSTEM LEVEL COMMAND                      *
' *                                                                   *
' *  GIVEN:     aCommand = the command to be executed                 *
' *                                                                   *
' *  RETURN:   nothing                                                *
' *                                                                   *
' *  NOTE:       Once the command has been issued, the statements that *
' *              follow the call to avExecute will be immediately     *
' *              executed, to pause ArcMap until the command is done,  *
' *              one possibility is to perform a loop checking for the *
' *              existence of a file that is created when the command *
' *              has finished processing, if this approach is suitable *
' *              use the subroutine avExecute2 rather than avExecute  *
' *                                                                   *
' *  Dim aCommand As String                                           *
' *                                                                   *
Public  Sub  avExecute2(aCommand,  aFileName)
' *                                                                   *
' *  PURPOSE:  TO EXECUTE A SYSTEM LEVEL COMMAND PAUSING ARCMAP UNTIL *
' *            A SPECIFIC FILE HAS BEEN FOUND                         *
' *                                                                   *
' *  GIVEN:     aCommand  = the command to be executed                *
' *             aFileName = the file to be searched for, until the    *
' *                         file is found this subroutine continues to *
' *                         process, once the file is found this      *
' *                         subroutine will terminate                 *
' *                                                                   *
' *  RETURN:   nothing                                                *
' *                                                                   *
' *  NOTE:       (a) Once the command has been issued, the statements *
' *                  that follow the call to avExecute2 will not be   *
' *                  executed until the file, aFileName, has been found *
' *              (b) In addition to waiting for the file, aFileName,  *
' *                  to exist, this subroutine will also terminate when *
' *                  a file whose name is the same as aFileName but has *
' *                  no file extension and is consistent in file size. *
' *                  For example, if aFileName is c:\temp\aFile.txt,  *
' *                  this subroutine will terminate if c:\temp\aFile is *
' *                  found and its file size is the same 10 consecutive *
' *                  times                                            *
' *              (c) The ArcMap document should have a name assigned to *
' *                  it other than the default of "Untitled...", if not *
' *                  it has been known for the Shell function to fail  *
' *                  for some reason, so assign a name to the document *
' *              (d) For Windows 2000 computers, this subroutine works *
' *                  much better than avExecute                       *
' *                                                                   *
' *  Dim aCommand As String, aFileName As String                      *
' *                                                                   *
Public  Sub  avFeatureInvalidate(pmxDoc  As  IMxDocument,  _
                                 theFeature As IFeature)
' *                                                                   *
' *  PURPOSE:  REDRAW A FEATURE                                       *
' *                                                                   *
' *  GIVEN:     pmxDoc     = the active view                          *
' *             theFeature = feature to be redrawn                    *
' *                                                                   *
' *  RETURN:   nothing                                                *
' *                                                                   *
' *  NOTE:       The theme or layer in which the feature resides in   *
' *              must have been made editable with avSetEditable prior *
' *              to calling this subroutine to ensure ugWrkSpcType is *
' *              properly defined                                     *
' *                                                                   *
' *  Dim pmxDoc As IMxDocument                                        *
' *  Dim theFeature As IFeature                                          *
' *                                                                     *
Public  Function  avFieldGetType(pField  As  iField)  As  esriFieldType
' *                                                                     *
' *   PURPOSE:   DETERMINE THE TYPE OF FIELD THAT A FIELD OBJECT IS     *
' *                                                                     *
' *   GIVEN:     pField          = field object to be processed         *
' *                                                                     *
' *   RETURN:    avFieldGetType  = numeric value denoting type of field *
' *                                0 : Small Integer                    *
' *                                1 : Long Integer                     *
' *                                2 : Single-precision float           *
' *                                3 : Double-precision float           *
' *                                4 : String                           *
' *                                5 : Date                             *
' *                                6 : Long Integer denoting the OID    *
' *                                7 : Geometry                         *
' *                                8 : Blob                             *
' *                                                                     *
' *  Dim pField As iField                                               *
' *  Dim avFieldGetType As esriFieldType                                *
' *                                                                     *
Public  Function  avFieldMake(aName,  aFieldType,  nChr,  ndr)  As  IFieldEdit
' *                                                                     *
' *   PURPOSE:   CREATE A FIELD THAT CAN BE ADDED TO A LAYER OR TABLE   *
' *                                                                     *
' *   GIVEN:     aName        = name of field to be created             *
' *              aFieldType   = type of field to be created as denoted  *
' *                             by the strings shown below on the left, *
' *                             to the right of these strings are the   *
' *                             field types that are actually created   *
' *                             BYTE          : Small Integer           *
' *                             CHAR          : String                  *
' *                             DATE          : Date                    *
' *                             DECIMAL       : Single                  *
' *                             DOUBLE        : Double                  *
' *                             FLOAT         : Single                  *
' *                             ISODATE       : Date                    *
' *                             ISODATETIME   : Date                    *
' *                             ISOTIME       : Date                    *
' *                             LOGICAL       : String                  *
' *                             LONG          : Integer                 *
' *                             MONEY         : Double                  *
' *                             SHORT         : Small Integer           *
' *                             BLOB          : Blob                    *
' *                             VCHAR         : String                  *
' *              nchr         = total character width of field including *
' *                             decimal point and negative sign, if they *
' *                             are to appear in the field              *
' *              ndr          = number of digits to the right of the    *
' *                             decimal point, 0 for non-numeric fields  *
' *                                                                     *
' *   RETURN:    avFieldMake = field object that was created            *
' *                                                                     *
' *   NOTE:      (a) This routine can not be used to create a geometry  *
' *                  field                                              *
' *              (b) If the name of the field exceeds 10 characters the *
' *                  field name that is created will contain only the   *
' *                  first 10 characters (use avSetAlias after the      *
' *                  table is created to assign the desired field name) *
' *                                                                     *
' *  Dim aName As String, aFieldType As String                          *
' *  Dim nchr As Long, ndr As Long                                      *
' *  Dim avFieldMake As IFieldEdit                                      *
' *                                                                     *
Public  Function  avFileCopy(nameFrom,  nameTo)
' *                                                                     *
' *   PURPOSE:   COPY A FILE FROM ONE LOCATION INTO ANOTHER             *
' *                                                                     *
' *   GIVEN:     nameFrom    = name of file to be copied                  *
' *               nameTo      = name of destination file                  *
' *                                                                       *
' *   RETURN:    avFileCopy = error flag (0 = no error, 1 = error)        *
' *                                                                       *
' *   NOTE:       (a) Wildcard characters can only be used in the last    *
' *                     path component of the nameFrom argument, that is:  *
' *                        nameFrom = "c:\mydocuments\letters\*.doc"       *
' *                      nameTo    = "c:\tempfolder\"                     *
' *                (b) The destination file will be overwritten if it      *
' *                   exists                                              *
' *                                                                       *
' *   Dim nameFrom As String, nameTo As String                            *
' *   Dim avFileCopy As Integer                                           *
' *                                                                       *
Public  Function  avFileDelete(name)
' *                                                                       *
' *   PURPOSE:  DELETE A FILE                                             *
' *                                                                       *
' *   GIVEN:     name           = name of file to be deleted, if the file *
' *                                 is not in the current folder a complete *
' *                                pathname must be specified             *
' *                                                                       *
' *   RETURN:    avFileDelete = error flag (0 = no error, 1 = error)      *
' *                                                                       *
' *   Dim name As String                                                  *
' *   Dim avFileDelete As Integer                                         *
' *                                                                       *
Public  Sub  avFileDialogPut(defname,  aPattrn,  Heading,  fileName)
' *                                                                       *
' *   PURPOSE:   TO CREATE A FILE USING A NAME THAT THE USER SPECIFIES    *
' *                                                                       *
' *   GIVEN:     defName   = default filename to be displayed             *
' *                aPattrn   = defines the pattern for similar files. Use  *
' *                              an asterisk as a wild card character, for *
' *                            example: *.ave  or *.*                     *
' *               Heading  = message box caption                         *
' *                                                                       *
' *   RETURN:    fileName = name of file to be created, if the user       *
' *                              Cancels the command, fileName will be set *
' *                             to a blank character (a single space)     *
' *                                                                       *
' *   Dim defName As String, aPattrn As String, Heading As String         *
' *   Dim fileName As String                                              *
' *                                                                       *
Public  Sub  avFileDialogReturnFiles(patrns,  labels,  heading,  defIndex,  _
                                        fileList)
' *                                                                       *
' *   PURPOSE:   GET A LIST OF FILE NAMES WHICH THE USER SELECTS          *
' *                                                                       *
' *   GIVEN:      patrns    = list of file patterns that can be displayed *
' *                labels    = list of labels corresponding to the list of *
' *                           patterns                                    *
' *               Heading  = message box caption                          *
' *                defIndex = index into pattern list denoting the        *
' *                             default pattern to be displayed           *
' *                                                                       *
' *   RETURN:    fileList = list of file names, of string type, that      *
' *                              were selected by the user, if the user   *
' *                              Cancels the command this list will be empty *
' *                                                                       *
' *   Dim patrns As New Collection, labels As New Collection              *
' *   Dim Heading As String, defIndex As Long                             *
' *   Dim fileList As New Collection                                      *
' *                                                                       *
Public  Function  avFileExists(name)  As  Boolean
' *                                                                       *
' *   PURPOSE:  DETERMINE IF A FILE EXISTS OR NOT                         *
' *                                                                       *
' *   GIVEN:     name           = name of file to be checked, if the file *
' *                               is not in the current folder a complete *
' *                               pathname must be specified              *
' *                                                                       *
' *   RETURN:    avFileExists = existence flag (true = yes, false = no)   *
' *                                                                       *
' *   Dim name As String                                                  *
' *   Dim avFileExists As Boolean                                         *
' *                                                                       *
Public  Function  avFindDoc(name)  As  Long
' *                                                                       *
' *   PURPOSE:   GET THE INDEX FOR A SPECIFIED LAYER OR TABLE NAME        *
' *                                                                       *
' *   GIVEN:     name       = name of theme or table to be found          *
' *                                                                       *
' *   RETURN:    avFindDoc = index into Table of Contents of the layer    *
' *                          or table, -1 if not found, index values      *
' *                          begin at 0 and increase sequentially by 1    *
' *                                                                       *
' *   NOTE:      (a) If the theme or table is found, the global           *
' *                  ugLayer or ugTable will be defined. If a theme is    *
' *                  found, ugLayer will be defined, if a table is        *
' *                  found ugTable will be defined. If nothing is found   *
' *                  both ugLayer and ugTable will be set to NOTHING      *
' *              (b) If the DLL version of Avenue Wraps is being used,    *
' *                  the properties Layer and Table should be used to     *
' *                  access the ugLayer and ugTable globals. That is,     *
' *                to process a layer:                                    *
' *                    aIndex = avFindDoc("LayerX")                       *
' *                    If(aIndex <> -1)Then                               *
' *                        Set pLayer = avwraps.Layer                     *
' *                    End If                                             *
' *                to process a table:                                    *
' *                    aIndex = avFindDoc("TableX")                       *
' *                    If(aIndex <> -1)Then                               *
' *                        Set pTable = avwraps.Table                     *
' *                    End If                                             *
' *              (c) The index value for a table will have the number     *
' *                  of layers added to it, so that we know a table is    *
' *                  being processed. The number of layers can be found   *
' *                  by using pMap.LayerCount, where pMap is an IMap      *
' *                object.                                                *
' *                                                                       *
' *   Dim name As Variant                                                 *
' *   Dim avFindDoc As Long                                               *
' *                                                                       *
Public  Function  avFTabExport(aTheme,  aFileName,  aClass,  selRecrds,  _
                               Optional addToDoc) As IFields
' *                                                                       *
' *   PURPOSE:   EXPORT A THEME TO CREATE A NEW THEME OR TABLE            *
' *                                                                       *
' *   GIVEN:     aTheme         = name of the theme to be exported        *
' *               aFileName     = name of the theme or table to be        *
' *                               created, if the name does not contain a *
' *                               complete pathname the current working   *
' *                               directory will be used, some examples   *
' *                             of name include:                          *
' *                                 c:\project\test\atheme                *
' *                                 c:\project\test\atheme.shp            *
' *                               atheme                                  *
' *                               atheme.shp                              *
' *                             the name can or can not contain the       *
' *                             extensions .dbf, .txt or .shp             *
' *               aClass        = type of theme or table to be created    *
' *                             dBase                                     *
' *                             TEXT                                      *
' *                             SHAPE                                     *
' *                selRecrds    = indicates if the selected features are  *
' *                             to be exported                            *
' *                              true   = export selected features only    *
' *                             false = export all features               *
' *              addToDoc      = optional argument indicating if the new *
' *                              theme or table is to be added to the     *
' *                             map (the default is true)                 *
' *                              true  = add new theme or table to map    *
' *                             false = do not add                        *
' *                                                                       *
' *   RETURN:    avFTabExport  = IFields object for the theme or table    *
' *                             that was created                          *
' *                                                                       *
' *   NOTE:      (a) if the theme or table to be created, aFileName,      *
' *                   exists on disk, it will be deleted before the       *
' *                    exporting is performed without informing the user  *
' *              (b) if the theme or table to be created is to be added *
' *                   to the map and it currently exists in the map, it   *
' *                   will be removed prior to deleting the existing      *
' *                   disk file(s) as well as performing the export       *
' *              (c) if selected features are to be exported and there    *
' *                   are no selected features, the entire theme will be *
' *                 exported                                              *
' *              (d) if the theme can not be exported for any reason      *
' *                   what so ever, avFTabExport will be set to NOTHING   *
' *              (e) if the new theme or table is not to be added to      *
' *                   the map, avFTabExport will be set to NOTHING        *
' *              (f) use the subroutine avInvalidateTOC to refresh the    *
' *                 Table of Contents                                     *
' *              (g) aTheme and aFileName can not be identical, they      *
' *                   must be different, if not an error is generated     *
' *                                                                       *
' *   Dim aTheme As String, aFileName As String, aClass As String         *
' *   Dim selRecrds As Boolean, addToDoc As Boolean                       *
' *   Dim avFTabExport As IFields                                         *
' *                                                                       *
Public  Function  avFTabMakeNew(aFileName,  aclass)  As  IFeatureLayer
' *                                                                       *
' *   PURPOSE:  CREATE A NEW SHAPEFILE                                    *
' *                                                                       *
' *   GIVEN:     aFileName      = name of the shapefile to be created,    *
' *                               if the name does not contain a          *
' *                                complete pathname the current working  *
' *                                directory will be used, some examples  *
' *                              of name include:                         *
' *                                  c:\project\test\l_0ln                *
' *                                  c:\project\test\l_0ln.shp            *
' *                                 l_0ln                                 *
' *                                 l_0ln.shp                             *
' *                               the name can or can not contain the     *
' *                              extension .shp                           *
' *              aClass         = type of shapefile to be created         *
' *                              POINT                                    *
' *                              MULTIPOINT                               *
' *                              POLYLINE                                 *
' *                              POLYGON                                  *
' *                              POINTM                                   *
' *                              MULTIPOINTM                              *
' *                              POLYLINEM                                *
' *                              POLYGONM                                 *
' *                              POINTZ                                   *
' *                              MULTIPOINTZ                              *
' *                              POLYLINEZ                                *
' *                              POLYGONZ                                 *
' *                                                                       *
' *   RETURN:    avFTabMakeNew = feature layer object that is created     *
' *                                                                       *
' *   NOTE:      (a) Three fields called FID, SHAPE and ID will be        *
' *                   created by this routine, the function avAddDoc can *
' *                   be used to add the shapefile to the map, if need    *
' *                 be                                                    *
' *              (b) If the shapefile to be created exists on disk, the *
' *                  routine will abort the existing shapefile will not *
' *                  be overwritten                                      *
' *                                                                      *
' *  Dim aFileName As String, aClass As String                           *
' *  Dim avFTabMakeNew As IFeatureLayer                                  *
' *                                                                      *
Public  Sub  avGetActiveDoc(pMxApp  As  IMxApplication,  _
                             pmxDoc As IMxDocument, _
                              pActiveView As IActiveView, pMap As IMap)
' *                                                                      *
' *  PURPOSE:  GET THE CURRENT DOCUMENT OR FOCUS MAP                     *
' *                                                                      *
' *  GIVEN:    nothing                                                   *
' *                                                                      *
' *  RETURN:   pMxApp      = the application                             *
' *             pmxDoc      = the document                               *
' *             pActiveView = the active view                            *
' *             pMap        = the focus map                              *
' *                                                                      *
' *  Dim pMxApp As IMxApplication, pmxDoc As IMxDocument                 *
' *  Dim pActiveView As IActiveView, pMap As IMap                        *
' *                                                                      *
Public  Sub  avGetActiveThemes(pmxDoc  As  IMxDocument,  ThemesList)
' *                                                                      *
' *  PURPOSE:  GET A LIST OF THE ACTIVE OR SELECTED THEMES               *
' *                                                                      *
' *  GIVEN:    pmxDoc     = the active view                              *
' *                                                                      *
' *  RETURN:   ThemesList = list of themes                               *
' *                                                                      *
' *  Dim pmxDoc As IMxDocument                                           *
' *  Dim ThemesList As New Collection                                    *
' *                                                                      *
Public  Function  avGetBaseName(aPath)  As  String
' *                                                                      *
' *  PURPOSE:  GET THE BASE NAME THAT APPEARS IN A PATH NAME             *
' *                                                                      *
' *  GIVEN:    aPath         = the full path name to be processed        *
' *                                                                      *
' *  RETURN:   avGetBaseName = base name appearing in a path name        *
' *                              including the filename extension, if    *
' *                             one is present in the base name          *
' *                                given                    return       *
' *                             c:\test\vb\aFile.shp         aFile.shp   *
' *                             c:\test\vb\aFile             aFile       *
' *                             c:\test\vb\                  vb          *
' *                             c:\test\vb                   vb          *
' *                             c:\a                         a           *
' *                             c:\                                      *
' *                             aFile.txt                    aFile.txt   *
' *                               Second from last example (c:\) yields  *
' *                             an empty string ("")                     *
' *                                                                      *
' *  Dim aPath As String                                                 *
' *  Dim avGetBaseName As String                                         *
' *                                                                      *
Public  Function  avGetBaseName2(aPath)  As  String
' *                                                                      *
' *  PURPOSE:  GET THE BASE NAME THAT APPEARS IN A PATH NAME MINUS       *
' *             ANY EXTENSION THAT MAY APPEAR IN THE BASE NAME           *
' *                                                                      *
' *  GIVEN:    aPath          = the full path name to be processed       *
' *                                                                      *
' *  RETURN:   avGetBaseName2 = base name appearing in a path name       *
' *                              without the filename extension          *
' *                                 given                    return      *
' *                              c:\test\vb\aFile.shp         aFile      *
' *                              c:\test\vb\aFile             aFile      *
' *                               c:\test\vb\                 vb        *
' *                               c:\test\vb                  vb        *
' *                               c:\a                        a         *
' *                               c:\                                   *
' *                               aFile.txt                   aFile     *
' *                                 Second from last example (c:\) yields *
' *                               an empty string ("")                  *
' *                                                                     *
' *  Dim aPath As String                                                *
' *  Dim avGetBaseName2 As String                                       *
' *                                                                     *
Public Sub avGetClassifications(thelegend As IFeatureRenderer, _
                                classList)
' *                                                                     *
' *  PURPOSE:  TO GET A LIST OF THE CLASSES USED IN A CLASSIFICATION    *
' *                                                                     *
' *  GIVEN:    theLegend = legend to be processed                       *
' *                                                                     *
' *  RETURN:   classList = list of classifications                      *
' *                                                                     *
' *  Dim theLegend As IFeatureRenderer                                  *
' *  Dim classList As New Collection                                    *
' *                                                                     *
Public Function avGetClassType(thelegend As IFeatureRenderer) As String
' *                                                                     *
' *  PURPOSE:  DETERMINE THE TYPE OF CLASSIFICATION THAT HAS BEEN       *
' *            USED TO CREATE A LEGEND                                  *
' *                                                                     *
' *  GIVEN:    theLegend      = legend to be processed                  *
' *                                                                     *
' *  RETURN:   avGetClassType = type of classification used in the      *
' *                               generation of a legend (renderer)     *
' *                               Manual (SingleSymbol and Unique)      *
' *                             DefinedInterval                         *
' *                             EqualInterval                           *
' *                             NaturalBreaks                           *
' *                             Quantile                                *
' *                             StandardDeviation                       *
' *                                                                     *
' *  Dim theLegend As IFeatureRenderer                                  *
' *  Dim avGetClassType As String                                       *
' *                                                                     *
Public Sub avGetDisplay(pActiveView As IActiveView, _
                        pScreenDisplay As IScreenDisplay, _
                        pDT As IDisplayTransformation)
' *                                                                     *
' *  PURPOSE:  GET THE CURRENT FOCUS MAP DISPLAY                        *
' *                                                                     *
' *  GIVEN:    pActiveView    = the focus map active view               *
' *                                                                     *
' *  RETURN:   pScreenDisplay = the screen display                      *
' *            pDT            = the screen display transformation       *
' *                                                                     *
' *  Dim pActiveView As IActiveView, pScreenDisplay As IScreenDisplay   *
' *  Dim pDT As IDisplayTransformation                                  *
' *                                                                     *
Public Sub avGetDisplayFlush()
' *                                                                     *
' *  PURPOSE:  SCRIPT TO MAKE SURE THE DISPLAY IS UP TO DATE BY         *
' *            FORCING ANY BUFFERED DRAWS TO BE DISPLAYED               *
' *                                                                     *
' *  GIVEN:    nothing                                                  *
' *                                                                     *
' *  RETURN:   nothing                                                  *
' *                                                                     *
Public Function avGetEnvVar(aPath)
' *                                                                     *
' *  PURPOSE:  GET THE FULL PATH FOR AN ENVIRONMENT VARIABLE            *
' *                                                                     *
' *   GIVEN:     aPath         = name of the environment variable name     *
' *                              to be processed                            *
' *                                                                         *
' *   RETURN:    avGetEnvVar  = full path associated with the variable      *
' *                                        given             return         *
' *                                        ARCHOME         C:\ARCGIS\ARCEXE81  *
' *                                          TMP           C:\WINDOWS\TEMP    *
' *                                         ABC                             *
' *                               The last example yields an empty string  *
' *                               (""), assuming the variable ABC does not *
' *                              exist                                      *
' *                                                                         *
' *  Dim aPath As String                                                    *
' *  Dim avGetEnvVar As String                                              *
' *                                                                         *
Public  Function  avGetExtension(aPath)  As  String
' *                                                                         *
' *   PURPOSE:   GET THE FILE EXTENSION IN A BASE NAME OF A PATH NAME       *
' *                                                                         *
' *   GIVEN:     aPath            = the full path name to be processed      *
' *                                                                         *
' *   RETURN:    avGetExtension = the filename extension                    *
' *                                        given                  return    *
' *                                  c:\test\vb\aFile.shp          shp      *
' *                                  c:\test\vb\aFile                       *
' *                                  c:\test\vb\                            *
' *                                  c:\test\vb                             *
' *                                  c:\a                                   *
' *                                  c:\                                    *
' *                                  aFile.shp                     shp      *
' *                                   Only the first and last examples      *
' *                                   yield non-empty strings ("")          *
' *                                                                         *
' *  Dim aPath As String                                                    *
' *  Dim avGetExtension As String                                           *
' *                                                                         *
Public  Function  avGetAlias(col)  As  String
' *                                                                         *
' *   PURPOSE:   GET THE ALIAS ASSIGNED TO A FIELD FOR A LAYER OR TABLE     *
' *                                                                         *
' *   GIVEN:     col          = index value representing the field for     *
' *                              which an alias is to be retrieved          *
' *                                                                         *
' *   RETURN:    avGetAlias = the alias assigned to the field               *
' *                                                                         *
' *   NOTE:       The current layer/table is processed, the subroutines    *
' *               avGetFTab or avGetVTab can be used to establish the      *
' *              current layer or table                                     *
' *                                                                         *
' *  Dim col As Long                                                        *
' *  Dim avGetAlias As String                                               *
' *                                                                         *
Public  Sub  avGetFeatData(pmxDoc  As  IMxDocument,  _
                   theTheme, theObjId, theFeature As IFeature, _
                    theShape As IGeometry, shapeType As esriGeometryType)
' *                                                                         *
' *   PURPOSE:   GET THE FEATURE DATA GIVEN A THEME AND AN OBJECT ID        *
' *                                                                         *
' *  GIVEN:     pmxDoc      = the active view                               *
' *              theTheme    = the theme to be processed                    *
' *              theObjId    = the object id of the desired feature         *
' *                                                                         *
' *  RETURN:    theFeature = the feature                                    *
' *              theShape    = the geometry of a feature                    *
' *              shapeType  = the shape type of a feature                   *
' *                                                                         *
' *  Dim pmxDoc As IMxDocument                                              *
' *  Dim theTheme As Variant, theObjId As Long                              *
' *  Dim theFeature As IFeature                                             *
' *  Dim theShape As IGeometry                                         *
' *  Dim shapeType As esriGeometryType                                 *
' *                                                                    *
Public Sub avGetFeature(pmxDoc As IMxDocument, _
                        theTheme, theObjId, theFeature As IFeature)
' *                                                                    *
' *  PURPOSE:  TO GET THE FEATURE GIVEN A THEME AND AN OBJECT ID       *
' *                                                                    *
' *  GIVEN:    pmxDoc     = the active view                            *
' *            theTheme   = the theme to be processed                  *
' *            theObjId   = the object id of the desired feature       *
' *                                                                    *
' *  RETURN:   theFeature = the feature                                *
' *                                                                    *
' *  NOTE:     (a) Use avGetTableRow if a table is to be processed     *
' *            (b) To get the geometry or shape of the feature use     *
' *                the following command after calling this routine    *
' *                     Set theShape = theFeature.Shape                *
' *                where theShape is an IGeometry object               *
' *                                                                    *
' *  Dim pmxDoc As IMxDocument                                         *
' *  Dim theTheme As Variant, theObjId As Long                         *
' *  Dim theFeature As IFeature                                        *
' *                                                                    *
Public Sub avGetFieldNames(thelegend As IFeatureRenderer, nameList)
' *                                                                    *
' *  PURPOSE:  GET THE FIELD NAMES USED TO CLASSIFY A THEME            *
' *                                                                    *
' *  GIVEN:    theLegend = legend to be processed                      *
' *                                                                    *
' *  RETURN:   nameList  = list of the field names that were used in   *
' *                        a classification (empty for SingleSymbol    *
' *                        legends or any other type of legend that    *
' *                        does not require a field name)              *
' *                                                                    *
' *  Dim theLegend As IFeatureRenderer                                 *
' *  Dim nameList As New Collection                                    *
' *                                                                    *
Public Sub avGetFields(theVTab As IFields, theList)
' *                                                                    *
' *  PURPOSE:  GET A LIST OF FIELD NAMES FOR A LAYER OR TABLE          *
' *                                                                    *
' *  GIVEN:    theVTab = field list for the theme or table             *
' *                                                                    *
' *  RETURN:   theList = list of field names for an attribute table,   *
' *                      these are not the alias names for the fields  *
' *                                                                    *
' *  Dim theVTab As IFields                                            *
' *  Dim theList As New Collection                                     *
' *                                                                    *
Public Sub avGetFTab(pmxDoc As IMxDocument, theTheme, _
                     theFTab As IFields, _
                     theFeatureClass As IFeatureClass, _
                     theLayer As IFeatureLayer)
' *                                                                    *
' *  PURPOSE:  GET THE ATTRIBUTE TABLE, FEATURE CLASS AND ASSOCIATED   *
' *            LAYER FOR A SPECIFIED THEME                             *
' *                                                                    *
' *  GIVEN:    pmxDoc          = the active view                       *
' *            theTheme        = the theme to be processed             *
' *                                                                    *
' *  RETURN:   theFTab         = the attribute table for the theme     *
' *            theFeatureClass = the feature class for the theme       *
' *            theLayer        = the associated layer for the theme    *
' *                                                                    *
' *  NOTE:     If a table, rather than a theme, is specified the       *
' *            theFeatureClass and theLayer will be set to Nothing     *
' *            while theFTab object will reflect the attributes for    *
' *            the table                                               *
' *                                                                    *
' *  Dim pmxDoc As IMxDocument                                         *
' *  Dim theTheme As Variant                                           *
' *  Dim theFTab As IFields, theFeatureClass As IFeatureClass          *
' *  Dim theLayer As IFeatureLayer                                     *
' *                                                                    *
Public Sub avGetFTabIDs(pmxDoc As IMxDocument, theTheme, _
                        theRecsList)
' *                                                                    *
' *  PURPOSE:  GET A LIST OF THE OBJECT IDS IN A LAYER                 *
' *                                                                    *
' *  GIVEN:    pmxDoc      = the active view                           *
' *            theTheme    = the theme to be processed                 *
' *                                                                    *
' *  RETURN:   theRecsList = the list of OIDs for the theme, this      *
' *                          list will include all OIDs for all of     *
' *                          the features in the theme                 *
' *                                                                    *
' *  Dim pmxDoc As IMxDocument                                         *
' *  Dim theTheme As Variant                                           *
' *  Dim theRecsList As New Collection                                 *
' *                                                                    *
Public Sub avGetFTabIDs2(pmxDoc As esriCore.IMxDocument, theTheme, _
                         theRecsArray)
' *                                                                    *
' *  PURPOSE:  BUILD AN ARRAY OF THE OBJECT IDS IN A LAYER             *
' *                                                                    *
' *  GIVEN:    pmxDoc       = the active view                          *
' *            theTheme     = the theme to be processed                *
' *                                                                    *
' *  RETURN:   theRecsArray = the array of OIDs for the theme, this    *
' *                           array will include all OIDs for all of   *
' *                           the features in the theme                *
' *                                                                    *
' *  NOTE:      (a) The first OID appears in the first element of the  *
' *                 array and can be accessed as shown below:          *
' *                      firstOID = theRecsArray(1)                    *
' *             (b) To determine the number of elements in the array   *
' *                 use the function, UBound, as shown below:          *
' *                      totalIDs = UBound(theRecsArray)               *
' *             (c) If the array can not be built, the number of       *
' *                 elements in the array will be one and the value of *
' *                 the first element in the array will be set to -1   *
' *             (d) Arrays process faster than lists, as such use this *
' *                 subroutine rather than avGetFTabIDs when the layer *
' *                 contains a large number of features                *
' *                                                                    *
' *  Dim pmxDoc As IMxDocument                                         *
' *  Dim theTheme As Variant                                           *
' *  Dim theRecsArray() As Long                                        *
' *                                                                    *
Public Sub avGetGeometry(pmxDoc As IMxDocument, _
                         theTheme, theObjId,theShape As IGeometry)
' *                                                                    *
' *  PURPOSE:  GET THE GEOMETRY OF A FEATURE GIVEN ITS THEME AND       *
' *            OBJECT ID                                               *
' *                                                                    *
' *  GIVEN:    pmxDoc   = the active view                              *
' *            theTheme = the theme to be processed                    *
' *            theObjId = the object id of the desired feature         *
' *                                                                    *
' *  RETURN:   theShape = the geometry of a feature                    *
' *                                                                    *
' *  Dim pmxDoc As IMxDocument                                         *
' *  Dim theTheme As Variant, theObjId As Long                         *
' *  Dim theShape As IGeometry                                         *
' *                                                                    *
```

```
Public  Sub  avGetGraphicList(pCurGraLyr  As  IGraphicsLayer,  graList)
' *                                                                       *
' *   PURPOSE:   TO GET A LIST OF THE GRAPHICS IN A GRAPHICS LAYER        *
' *                                                                       *
' *   GIVEN:      pCurGraLyr = the graphics layer containing the user     *
' *                              programmed graphics                      *
' *                                                                       *
' *   RETURN:     graList      = list of graphic elements in the graphics *
' *                              layer                                    *
' *                                                                       *
' *   Dim pCurGraLyr As IGraphicsLayer                                    *
' *   Dim graList As New Collection                                       *
' *                                                                       *
Public  Function  avGetLayerIndx(theTheme)  As  Long
' *                                                                       *
' *   PURPOSE:   TO DETERMINE THE INDEX OF THE LAYER OR TABLE WE ARE      *
' *              DEALING WITH                                             *
' *                                                                       *
' *   GIVEN:      theTheme        = the layer or table to be processed    *
' *                                                                       *
' *   RETURN:     avGetLayerIndx = index of the layer or table            *
' *                                                                       *
' *   Dim theTheme As Variant                                             *
' *   Dim avGetLayerIndx As Long                                          *
' *                                                                       *
Public  Sub  avGetLayerType(pSelected  As  IUnknown,  aName,  aType)
' *                                                                       *
' *   PURPOSE:   TO DETERMINE THE TYPE OF LAYER WE ARE DEALING WITH       *
' *                                                                       *
' *   GIVEN:      pSelected = the data layer object to be processed       *
' *                                                                       *
' *   RETURN:     aName       = name of the data layer object (uppercase) *
' *               aType      = type of data layer object                  *
' *                             0 = unknown                               *
' *                             1 = standalone table                      *
' *                             2 = raster layer                          *
' *                             3 = tin layer                             *
' *                             4 = annotation layer                      *
' *                             5 = feature layer                         *
' *                             6 = CAD annotation layer                  *
' *                             7 = CAD layer                             *
' *                                                                       *
' *   NOTE:        This subroutine will generate an error message if the  *
' *                 object that is passed in is a standalone featureclass *
' *                 in an SDE geodatabase.  The solution at this time is  *
' *                 to put the standalone featureclass into a dataset in  *
' *               the SDE geodatabase.                                    *
' *                                                                       *
' *   Dim pSelected As IUnknown                                           *
' *   Dim aName As Variant, aType As Integer                              *
' *                                                                       *
Public  Sub  avGetLegend(pmxDoc  As  IMxDocument,  theTheme,  _
                         aLegend As IFeatureRenderer)
' *                                                                       *
' *   PURPOSE:   TO GET THE LEGEND THAT IS ASSOCIATED WITH A THEME        *
' *                                                                       *
' *   GIVEN:     pmxDoc    = the active view                              *
' *              theTheme = theme to be processed                         *
' *                                                                       *
' *   RETURN:    aLegend  = theme legend                                  *
' *                                                                       *
' *   Dim pmxDoc As IMxDocument                                           *
' *   Dim theTheme As Variant                                             *
' *   Dim aLegend As IFeatureRenderer                                     *
' *                                                                       *
Public  Function  avGetLegendType(thelegend  _
                                  As IFeatureRenderer) As String
' *                                                                       *
' *   PURPOSE:   DETERMINE THE TYPE OF LEGEND IN USE                      *
' *                                                                    *
' *  GIVEN:     theLegend        = legend to be processed              *
' *                                                                    *
' *  RETURN:    avGetLegendType = type of legend (renderer) in use     *
' *                               UNIQUE       : Unique Value          *
' *                               SIMPLE       : Simple                *
' *                               SCALE        : ScaleDependent        *
' *                               PROPORTIONAL : Proportional Symbol   *
' *                               BIVARIATE    : Bivariate             *
' *                               CHART        : Chart                 *
' *                               CLASS        : Class Breaks          *
' *                               DOT          : Dot Density           *
' *                                                                    *
' *  Dim theLegend As IFeatureRenderer                                 *
' *  Dim avGetLegendType As String                                     *
' *                                                                    *
Public Function avGetMaxOID(pmxdoc As IMxDocument, theTheme) As Long
' *                                                                    *
' *  PURPOSE:  GET THE LARGEST OID IN A LAYER OR TABLE                 *
' *                                                                    *
' *  GIVEN:     pmxDoc      = the active view                          *
' *             theTheme    = the theme or table to be processed       *
' *                                                                    *
' *  RETURN:    avGetMaxOID = largest OID in the theme or table        *
' *                                                                    *
' *  Dim pmxDoc As IMxDocument                                         *
' *  Dim theTheme As Variant                                           *
' *  Dim avGetMaxOID As Long                                           *
' *                                                                    *
Public Sub avGetName(aTitle)
' *                                                                    *
' *  PURPOSE:  TO GET THE CAPTION OF THE APPLICATION                   *
' *                                                                    *
' *  GIVEN:     nothing                                                *
' *                                                                    *
' *  RETURN:    aTitle  = name of the application appearing in the     *
' *                       upper left corner of the application window  *
' *                                                                    *
' *  Dim aTitle As String                                              *
' *                                                                    *
Public Function avGetNumClasses(thelegend As IFeatureRenderer) As Long
' *                                                                    *
' *  PURPOSE:  DETERMINE THE NUMBER OF CLASSES IN A LEGEND             *
' *                                                                    *
' *  GIVEN:     theLegend        = legend to be processed              *
' *                                                                    *
' *  RETURN:    avGetNumClasses = number of classes in the legend      *
' *                                                                    *
' *  Dim theLegend As IFeatureRenderer                                 *
' *  Dim avGetNumClasses As Long                                       *
' *                                                                    *
Public Function avGetNumRecords(pmxDoc As IMxDocument, theTheme) As Long
' *                                                                    *
' *  PURPOSE:  GET THE NUMBER OF RECORDS IN A LAYER OR TABLE           *
' *                                                                    *
' *  GIVEN:     pmxDoc          = the active view                      *
' *             theTheme        = the theme or table to be processed   *
' *                                                                    *
' *  RETURN:    avGetNumRecords = number of records in theme or table  *
' *                                                                    *
' *  Dim pmxDoc As IMxDocument                                         *
' *  Dim theTheme As Variant                                           *
' *  Dim avGetNumRecords As Long                                       *
' *                                                                    *
Public Function avGetPalette(pmxDoc As IMxDocument) As IStyleGallery
' *                                                                    *
' *  PURPOSE:  GET THE VARIOUS PALETTES AVAILABLE IN THE APPLICATION   *
' *                                                                    *
' *  GIVEN:     pmxDoc       = the active view                         *
' *                                                                       *
' *  RETURN:    avGetPalette = the available style galleries              *
' *                             Reference Systems :  12 items             *
' *                             Shadows           :  12 items             *
' *                             Area Patches      :   8 items             *
' *                             Line Patches      :   9 items             *
' *                             Labels            :  20 items             *
' *                             North Arrows      :  97 items             *
' *                             Scale Bars        :  11 items             *
' *                             Legend Items      :  18 items             *
' *                             Scale Text        :   7 items             *
' *                             Color Ramps       :  78 items             *
' *                             Borders           :  16 items             *
' *                             Backgrounds       :  18 items             *
' *                             Colors            : 120 items             *
' *                             Fill Symbols      :  53 items             *
' *                             Line Symbols      :  86 items             *
' *                             Marker Symbols    : 114 items             *
' *                             Text Symbols      :  35 items             *
' *                                                                       *
' *  NOTE:      Depending upon the installation, the total number of      *
' *             galleries and items within each gallery may vary          *
' *                                                                       *
' *  Dim pmxDoc As IMxDocument                                            *
' *  Dim avGetPalette As IStyleGallery                                    *
' *                                                                       *
Public  Function  avGetPathName(aPath)  As  String
' *                                                                       *
' *  PURPOSE:  GET THE PATH NAME THAT APPEARS IN A STRING                 *
' *                                                                       *
' *  GIVEN:     aPath           = the full path name to be processed      *
' *                                                                       *
' *  RETURN:    avGetPathName  = path name appearing in a string minus    *
' *                               the last component in the string        *
' *                                  given                  return        *
' *                               c:\test\vb\aFile.shp       c:\test\vb   *
' *                               c:\test\vb\aFile           c:\test\vb   *
' *                              c:\test\vb\                c:\test       *
' *                              c:\test\vb                 c:\test       *
' *                             c:\a                        c:\           *
' *                             c:\                                       *
' *                             aFile.shp                                 *
' *                               The last two examples will yield empty *
' *                             strings ("")                              *
' *                                                                       *
' *  Dim aPath As String                                                  *
' *  Dim avGetPathName As String                                          *
' *                                                                       *
Public  Function  avGetPrecision(theFTab  As  IFields,  theField)  As  Long
' *                                                                       *
' *  PURPOSE:  GET THE PRECISION OF A FIELD                               *
' *                                                                       *
' *  GIVEN:     theFTab         = the FTab or VTab to be processed        *
' *               theField        = index into FTab or VTab representing  *
' *                                the field to be processed              *
' *                                                                       *
' *  RETURN:    avGetPrecision = number of digits to the right of the     *
' *                               decimal point                           *
' *                                                                       *
' *  NOTE:       This function always returns 0 for fields contained in *
' *             a personal geodatabase                                    *
' *                                                                       *
' *  Dim theFTab As IFields, theField As Long                             *
' *  Dim avGetPrecision As Long                                           *
' *                                                                       *
Public  Sub  avGetProjectName(aTitle)
' *                                                                       *
' *  PURPOSE:  GET THE NAME OF THE CURRENT DOCUMENT                       *
' *                                                                       *
' *  GIVEN:    nothing                                                *
' *                                                                   *
' *   RETURN:    aTitle  = the name of the current document, this will  *
' *                          include the .mxd extension (i.e. sample.mxd) *
' *                                                                   *
' *  Dim aTitle As String                                             *
' *                                                                   *
Public  Sub  avGetSelected(pmxDoc  As  IMxDocument,  selGraphList)
' *                                                                   *
' *   PURPOSE:  GET THE SELECTED GRAPHIC TEXT IN THE MAP              *
' *                                                                   *
' *  GIVEN:    pmxDoc       = the active view                         *
' *                                                                   *
' *   RETURN:    selGraphList =  list containing the selected graphic  *
' *                              text elements                         *
' *                                                                   *
' *  Dim pmxDoc As IMxDocument                                        *
' *  Dim selGraphList As New Collection                               *
' *                                                                   *
Public  Sub  avGetSelectedExtent(pmxDoc  As  IMxDocument,  _
                                   theTheme, theRect As IEnvelope)
' *                                                                   *
' *   PURPOSE:  GET THE ENCLOSING RECTANGLE FOR THE SELECTED SET OF A  *
' *               THEME, OR THE ENCLOSING RECTANGLE FOR THE ENTIRE THEME *
' *               IF THE SELECTED SET IS EMPTY (NO SELECTED FEATURES)  *
' *                                                                   *
' *  GIVEN:    pmxDoc   = the active view                             *
' *             theTheme = the theme to be processed                   *
' *                                                                   *
' *   RETURN:    theRect   = the enclosing rectangle encompassing the  *
' *                          selected features, or all of the features, *
' *                         if no features are selected                *
' *                                                                   *
' *  Dim pmxDoc As IMxDocument                                        *
' *  Dim theTheme As Variant                                          *
' *  Dim theRect As IEnvelope                                         *
' *                                                                   *
Public  Sub  avGetSelection(pmxDoc  As  IMxDocument,  theTheme,  _
                              psTableSel As ISelectionSet)
' *                                                                   *
' *   PURPOSE:  GET THE SELECTED SET FOR A FOR A LAYER OR TABLE       *
' *                                                                   *
' *  GIVEN:    pmxDoc      = the active view                          *
' *              theTheme    = the theme or table to be processed      *
' *                                                                   *
' *   RETURN:   psTableSel = the selection set for the theme           *
' *                                                                   *
' *  Dim pmxDoc As IMxDocument                                        *
' *  Dim theTheme As Variant                                          *
' *  Dim psTableSel As ISelectionSet                                  *
' *                                                                   *
Public  Sub  avGetSelectionClear(pmxDoc  As  IMxDocument,  theTheme,  theRcrd)
' *                                                                   *
' *   PURPOSE:  REMOVE A RECORD FROM THE SELECTED SET FOR A LAYER OR  *
' *             TABLE                                                 *
' *                                                                   *
' *  GIVEN:    pmxDoc   = the active view                             *
' *              theTheme = the theme or table to be processed         *
' *               theRcrd  = the record to be removed from the selection *
' *                                                                   *
' *  RETURN:   nothing                                                *
' *                                                                   *
' *  Dim pmxDoc As IMxDocument                                        *
' *  Dim theTheme As Variant                                          *
' *  Dim theRcrd As Long                                              *
' *                                                                   *
Public  Sub  avGetSelectionIDs(psTableSel  As  ISelectionSet,  selRecsList)
' *                                                                   *
' *   PURPOSE:  BUILD A COLLECTION OF OIDS FROM A SELECTION SET       *
' *                                                                    *
' *   GIVEN:      psTableSel   = the selection set for a theme or table *
' *                                                                    *
' *   RETURN:     selRecsList  = the list of OIDs for the selection set *
' *                                                                    *
' *   Dim psTableSel As ISelectionSet                                  *
' *   Dim selRecsList As New Collection                                *
' *                                                                    *
Public Sub avGetSelectionIDs2(psTableSel As esriCore.ISelectionSet, _
                              selRecsArray)
' *                                                                    *
' *   PURPOSE:   BUILD AN ARRAY OF OIDS FROM A SELECTION SET           *
' *                                                                    *
' *   GIVEN:      psTableSel    = the selection set for a theme or table *
' *                                                                    *
' *   RETURN:     selRecsArray = the array containing the OIDs for the *
' *                              selection set                         *
' *                                                                    *
' *   NOTE:       (a) The first OID appears in the first element of the *
' *                   array and can be accessed as shown below:        *
' *                           firstOID = selRecsArray(1)               *
' *               (b) To determine the number of elements in the array *
' *                   use the function, UBound, as shown below:        *
' *                           totalIDs = UBound(selRecsArray)          *
' *               (c) If the array can not be built, the number of     *
' *                   elements in the array will be one and the value of *
' *                   the first element in the array will be set to -1 *
' *               (d) Arrays process faster than lists, as such use this *
' *                   subroutine rather than avGetSelectionIDs when the *
' *                   selection set is large                           *
' *                                                                    *
' *   Dim psTableSel As ISelectionSet                                  *
' *   Dim selRecsArray() As Long                                       *
' *                                                                    *
Public Sub avGetSelFeatures(pmxDoc As IMxDocument, _
                            themeList, selMode, selThmList, selRecList)
' *                                                                    *
' *   PURPOSE:   PRESERVE THE THEMES AND RECORD NUMBERS OF THEMES WITH *
' *              SELECTED FEATURES                                     *
' *                                                                    *
' *   GIVEN:      pmxDoc      = the active view                        *
' *               themeList   = list of themes to be processed         *
' *               selMode     = mode of selection                      *
' *                             0  = all features                      *
' *                             1  = point features                    *
' *                             2  = polyline features                 *
' *                             3  = polygon features                  *
' *                             4  = polyline and polygon features     *
' *                             10 = same as 0 and include themes that *
' *                                  do not have selected features     *
' *                                                                    *
' *   RETURN:     selThmList = list of themes with selected features   *
' *               selRecList = list of selected features record numbers *
' *                                                                    *
' *   NOTE:       (a) structure of selThmList is:                      *
' *                   Item 1: name of theme 1                          *
' *                   Item 2: number of selected features in theme 1   *
' *                   Item 3: name of theme 2                          *
' *                   Item 4: number of selected features in theme 2   *
' *                   Repeat Items 1 and 2 for each theme              *
' *               (b) structure of selRecList is:                      *
' *                   Item 1: selected feature 1 OID in theme 1        *
' *                   Item 2: selected feature 2 OID in theme 1        *
' *                   Repeat Item 1 for each selected feature in theme 1 *
' *                   Item 3: selected feature 1 OID in theme 2        *
' *                   Item 4: selected feature 2 OID in theme 2        *
' *                   Repeat Item 3 for each selected feature in theme 2 *
' *                                                                    *
' *   Dim pmxDoc As IMxDocument                                        *
' *   Dim themeList As New Collection, selMode As Integer         *
' *   Dim selThmList As New Collection, selRecList As New Collection    *
' *                                                                 *
Public Function avGetShapeType(pmxDoc As IMxDocument, _
                                    theTheme) As esriGeometryType
' *                                                                 *
' *   PURPOSE:  GET THE DEFAULT SHAPE TYPE FOR A THEME              *
' *                                                                 *
' *   GIVEN:    pmxDoc          = the active view                   *
' *             theTheme        = the theme to be processed         *
' *                                                                 *
' *   RETURN:   avGetShapeType = the default shape type             *
' *                                                                 *
' *   Dim pmxDoc As IMxDocument                                     *
' *   Dim theTheme As Variant                                       *
' *   Dim avGetShapeType As esriGeometryType                        *
' *                                                                 *
Public Sub avGetTableRow(pmxdoc As IMxDocument, theTable, _
                              theObjId, theRow As IRow)
' *                                                                 *
' *   PURPOSE:  GET THE ROW GIVEN A TABLE AND AN OBJECT ID OF A RECORD *
' *             IN THE TABLE                                        *
' *                                                                 *
' *   GIVEN:    pmxDoc   = the active view                          *
' *             theTable = the table to be processed                *
' *             theObjId = the object id of the desired record      *
' *                                                                 *
' *   RETURN:   theRow   = the row                                  *
' *                                                                 *
' *   NOTE:     Use avGetFeature if a theme is to be processed      *
' *                                                                 *
' *   Dim pmxDoc As IMxDocument                                     *
' *   Dim theTable As Variant, theObjId As Long                     *
' *   Dim theRow As IRow                                            *
' *                                                                 *
Public Sub avGetTables(pmxdoc As IMxDocument, nameList, tableList)
' *                                                                 *
' *   PURPOSE:  GET A LIST OF THE TABLES IN THE DOCUMENT            *
' *                                                                 *
' *   GIVEN:    pmxDoc    = the active view                         *
' *                                                                 *
' *   RETURN:   nameList  = list of names for the tables which exist  *
' *                         in the document                         *
' *             tableList = list of ITable objects that were found  *
' *                                                                 *
' *   NOTE:     The number of items in nameList and tableList will be  *
' *             the same. The items in nameList simply reflect the  *
' *             names of the ITable objects in tableList            *
' *                                                                 *
' *   Dim pmxDoc As IMxDocument                                     *
' *   Dim nameList As New Collection                                *
' *   Dim tableList As New Collection                               *
' *                                                                 *
Public Sub avGetThemeExtent(pmxDoc As IMxDocument, theTheme, _
                                 theRect As IEnvelope)
' *                                                                 *
' *   PURPOSE:  GET THE ENCLOSING RECTANGLE FOR A THEME             *
' *                                                                 *
' *   GIVEN:    pmxDoc   = the active view                          *
' *             theTheme = the theme to be processed                *
' *                                                                 *
' *   RETURN:   theRect  = the enclosing rectangle encompassing all  *
' *                        of the features in the theme             *
' *                                                                 *
' *   Dim pmxDoc As IMxDocument                                     *
' *   Dim theTheme As Variant                                       *
' *   Dim theRect As IEnvelope                                      *
' *                                                                 *
```

```
Public Sub avGetThemes(pmxDoc As IMxDocument, opmode, ThemesList)
' *                                                                      *
' *  PURPOSE:  GET A LIST OF THEMES OR TABLES                            *
' *                                                                      *
' *  GIVEN:    pmxDoc     = the active view                              *
' *             opmode     = mode of operation                           *
' *                          0 = find all layers                         *
' *                            1 = find only feature & annotation layers *
' *                          2 = find all tables                         *
' *                           3 = find only annotation layers            *
' *                           4 = find only feature layers               *
' *                            5 = same as 0 except expand group layers  *
' *                                                                      *
' *  RETURN:   ThemesList = list of themes                               *
' *                                                                      *
' *  Dim pmxDoc As IMxDocument                                           *
' *  Dim opmode As Integer                                               *
' *  Dim ThemesList As New Collection                                    *
' *                                                                      *
Public Sub avGetUniqueValues(pmxDoc As IMxDocument, _
                             theTheme, aField, aList)
' *                                                                      *
' *  PURPOSE:  GET A LIST OF UNIQUE VALUES FOR A FIELD IN A THEME OR     *
' *            A TABLE                                                   *
' *                                                                      *
' *  GIVEN:    pmxDoc   = the active view                                *
' *              theTheme = name of the theme or table to be processed   *
' *             aField   = name of the field to be processed             *
' *                                                                      *
' *  RETURN:   aList     = list of unique values for the specified       *
' *                        field                                         *
' *                                                                      *
' *  Dim pmxDoc As IMxDocument                                           *
' *  Dim theTheme As Variant, aField As String                           *
' *  Dim aList As New Collection                                         *
' *                                                                      *
Public Sub avGetVisibleCADLayers(pmxDoc As IMxDocument, vThemesList)
' *                                                                      *
' *  PURPOSE:  GET A LIST OF THE VISIBLE CAD ANNOTATION LAYERS AND       *
' *            VISIBLE CAD LAYERS                                        *
' *                                                                      *
' *  GIVEN:    pmxDoc      = the active view                             *
' *                                                                      *
' *  RETURN:   vThemesList = list of visible CAD layers                  *
' *                                                                      *
' *  Dim pmxDoc As IMxDocument                                           *
' *  Dim vThemesList As New Collection                                   *
' *                                                                      *
Public Sub avGetVisibleThemes(pmxDoc As IMxDocument, vThemesList)
' *                                                                      *
' *  PURPOSE:  GET A LIST OF THE VISIBLE THEMES                          *
' *                                                                      *
' *  GIVEN:    pmxDoc      = the active view                             *
' *                                                                      *
' *  RETURN:   vThemesList = list of visible themes                      *
' *                                                                      *
' *  NOTE:       (a) Only annotation and feature layers are processed    *
' *              (b) The layer scale threshold must be satisfied, in     *
' *                  addition to the layer being visibile, in order      *
' *                  for the layer to be determined as being visible     *
' *                                                                      *
' *  Dim pmxDoc As IMxDocument                                           *
' *  Dim vThemesList As New Collection                                   *
' *                                                                      *
Public Sub avGetVTab(pmxDoc As IMxDocument, theTheme, _
                        theVTab As IFields)
' *                                                                      *
' *  PURPOSE:  GET THE ATTRIBUTE TABLE FOR A LAYER OR TABLE              *
' *                                                                      *
' *  GIVEN:     pmxDoc    = the active view                               *
' *               theTheme = the theme or table to be processed           *
' *                                                                       *
' *   RETURN:    theVTab   = the attribute table for the theme or table   *
' *                                                                       *
' *  Dim pmxDoc As IMxDocument                                            *
' *  Dim theTheme As Variant                                              *
' *  Dim theVTab As IFields                                               *
' *                                                                       *
Public Sub avGetVTabIDs(pmxDoc As IMxDocument, theTable, theRecsList)
' *                                                                       *
' *  PURPOSE:  GET A LIST OF OBJECT IDS FOR A TABLE                       *
' *                                                                       *
' *  GIVEN:     pmxDoc       = the active view                            *
' *              theTable     = the table to be processed                 *
' *                                                                       *
' *  RETURN:    theRecsList = the list of OIDs for the table              *
' *                                                                       *
' *  Dim pmxDoc As IMxDocument                                            *
' *  Dim theTable As Variant                                              *
' *  Dim theRecsList As New Collection                                    *
' *                                                                       *
Public Sub avGetVTabIDs2(pmxDoc As esriCore.IMxDocument, theTable, _
                          theRecsArray)
' *                                                                       *
' *  PURPOSE:  BUILD AN ARRAY OF OBJECT IDS FOR A TABLE                   *
' *                                                                       *
' *  GIVEN:     pmxDoc        = the active view                           *
' *              theTable      = the table to be processed                *
' *                                                                       *
' *  RETURN:    theRecsArray = the array of OIDs for the table            *
' *                                                                       *
' *  NOTE:       (a) The first OID appears in the first element of the   *
' *                   array and can be accessed as shown below:           *
' *                          firstOID = theRecsArray(1)                   *
' *                (b) To determine the number of elements in the array   *
' *                   use the function, UBound, as shown below:           *
' *                          totalIDs = UBound(theRecsArray)              *
' *               (c) If the array can not be built, the number of        *
' *                    elements in the array will be one and the value of *
' *                    the first element in the array will be set to -1   *
' *                (d) Arrays process faster than lists, as such use this *
' *                     subroutine rather than avGetVTabIDs when the table *
' *                   contains a large number of records                  *
' *                                                                       *
' *  Dim pmxDoc As IMxDocument                                            *
' *  Dim theTable As Variant                                              *
' *  Dim theRecsArray() As Long                                           *
' *                                                                       *
Public Sub avGetWinFonts(aColl)
' *                                                                       *
' *  PURPOSE:  GET A LIST OF THE FONTS INSTALLED ON THE PC                *
' *                                                                       *
' *  GIVEN:     nothing                                                   *
' *                                                                       *
' *   RETURN:    aColl    = an alphabetically sorted list of the fonts    *
' *                          that are installed on the computer           *
' *                                                                       *
' *  Dim aColl As New Collection                                          *
' *                                                                       *
Public Sub avGetWorkDir(theWorkDir)
' *                                                                       *
' *  PURPOSE:  GET THE CURRENT WORKING DIRECTORY                          *
' *                                                                       *
' *  GIVEN:     nothing                                                   *
' *                                                                       *
' *  RETURN:    theWorkDir = current working directory                    *
' *                                                                       *
' *  Dim theWorkDir As String                                             *
' *                                                                      *
Public  Function  avGraphicGetShape(pElement  As  IElement)  As  IGeometry
' *                                                                      *
' *   PURPOSE:   TO GET THE GEOMETRY THAT IS ASSOCIATED WITH A GRAPHIC   *
' *                                                                      *
' *  GIVEN:     pElement           = the graphic to be processed         *
' *                                                                      *
' *   RETURN:    avGraphicGetShape = geometry describing the graphic     *
' *                                                                      *
' *   NOTE:      This routine will process PEN, MARKER, FILL and TEXT    *
' *               symbols since they all share the IElement interface    *
' *               For TEXT symbols the geometry passed back will be that *
' *               of an inclined rectangle that encloses the text string *
' *                                                                      *
' *  Dim pElement As IElement                                            *
' *  Dim avGraphicGetShape As IGeometry                                  *
' *                                                                      *
Public  Function  avGraphicGetSymbol(aSymTyp,  _
                                     pElement As IElement) As ISymbol
' *                                                                      *
' *   PURPOSE:   TO GET THE SYMBOL THAT IS ASSOCIATED WITH A GRAPHIC     *
' *                                                                      *
' *  GIVEN:     aSymTyp              = type of graphic to be assigned    *
' *                                     PEN    : line symbol             *
' *                                     MARKER : point symbol            *
' *                                     FILL   : polygon symbol          *
' *              pElement            = the graphic to be processed       *
' *                                                                      *
' *   RETURN:    avGraphicGetSymbol = symbol describing the graphic      *
' *                                     element, its properties such as  *
' *                                    its color                         *
' *                                                                      *
' *  Dim aSymTyp As String, pElement As IElement                         *
' *  Dim avGraphicGetSymbol As ISymbol                                   *
' *                                                                      *
Public  Sub  avGraphicInvalidate(pElement  As  IElement)
' *                                                                      *
' *  PURPOSE:   TO REDRAW OR UPDATE A GRAPHIC ELEMENT                    *
' *                                                                      *
' *   GIVEN:      pElement = graphic to be redrawn due to a change that  *
' *                          has been made to it                         *
' *                                                                      *
' *  RETURN:   nothing                                                   *
' *                                                                      *
' *   NOTE:      Use this routine to redraw a graphic that has already   *
' *               been added to a graphics layer and subsequently        *
' *             modified                                                 *
' *                                                                      *
' *  Dim pElement As IElement                                            *
' *                                                                      *
Public  Sub  avGraphicListDelete(pCurGraLyr  As  IGraphicsLayer)
' *                                                                      *
' *  PURPOSE:  TO DELETE A GRAPHICS LAYER                                *
' *                                                                      *
' *   GIVEN:      pCurGraLyr = the graphics layer containing the user    *
' *                            programmed graphics                       *
' *                                                                      *
' *  RETURN:   nothing                                                   *
' *                                                                      *
' *   NOTE:      The basic graphics layer can not be deleted, only user *
' *               created graphics layers can be deleted with this macro *
' *                                                                      *
' *  Dim pCurGraLyr As IGraphicsLayer                                    *
' *                                                                      *
Public  Sub  avGraphicListEmpty(pCurGraLyr  As  IGraphicsLayer)
' *                                                                      *
' *   PURPOSE:  TO DELETE THE GRAPHICS FROM A GRAPHICS LAYER             *
' *                                                                      *
' *   GIVEN:      pCurGraLyr = the graphics layer containing the user    *
' *                            programmed graphics                       *
' *                                                                      *
' *  RETURN:   nothing                                                   *
' *                                                                      *
' *  NOTE:       The graphics within the graphics layer is deleted, the *
' *              graphics layer itself is not deleted so that graphics  *
' *              can be added to the graphics layer at another time     *
' *                                                                      *
' *  Dim pCurGraLyr As IGraphicsLayer                                    *
' *                                                                      *
Public  Sub  avGraphicSetShape(pElement  As  IElement,  _
                                   theGeom As IGeometry)
' *                                                                      *
' *  PURPOSE:  TO SET THE GEOMETRY THAT IS ASSOCIATED WITH A GRAPHIC    *
' *                                                                      *
' *  GIVEN:     pElement = the graphic to be modified                    *
' *               theGeom  = the new geometry that describes the graphic *
' *                                                                      *
' *  RETURN:   nothing                                                   *
' *                                                                      *
' *  NOTE:      The geometry of the given element is modified by this   *
' *              routine, so that pElement is different after calling   *
' *             this routine                                             *
' *                                                                      *
' *  Dim pElement As IElement, theGeom As IGeometry                      *
' *                                                                      *
Public  Sub  avGraphicSetSymbol(aSymTyp,  _
                                    pElement As IElement, pSymbol As ISymbol)
' *                                                                      *
' *  PURPOSE:  TO ASSIGN A SYMBOL TO A GRAPHIC                           *
' *                                                                      *
' *  GIVEN:     aSymTyp  = type of graphic to be assigned                *
' *                          PEN    : line symbol                        *
' *                          MARKER : point symbol                       *
' *                          FILL   : polygon symbol                     *
' *              pElement = graphic for which the given symbol is to be *
' *                          assigned to                                 *
' *             pSymbol  = symbol to be assigned to the graphic          *
' *                                                                      *
' *  RETURN:   nothing                                                   *
' *                                                                      *
' *   Dim aSymTyp As String, pElement As IElement, pSymbol As ISymbol   *
' *                                                                      *
Public  Function  avGraphicShapeMake(aSymTyp,  _
                                         theGeom As IGeometry) As IElement
' *                                                                      *
' *  PURPOSE:  TO CREATE A GRAPHIC SHAPE THAT CAN BE ADDED TO THE        *
' *            GRAPHICS LIST                                             *
' *                                                                      *
' *  GIVEN:     aSymTyp              = type of graphic to be created     *
' *                                    PEN    : line symbol              *
' *                                    MARKER : point symbol             *
' *                                    FILL   : polygon symbol           *
' *             theGeom              = geometry describing the graphic   *
' *                                                                      *
' *  RETURN:    avGraphicShapeMake = the graphic that can be added to   *
' *                                    the graphics layer                *
' *                                                                      *
' *  Dim aSymTyp As String, theGeom As IGeometry                         *
' *  Dim avGraphicShapeMake As IElement                                  *
' *                                                                      *
Public  Function  avGraphicTextGetAngle(aGraphicText  _
                                           As IElement) As Double
' *                                                                      *
' *  PURPOSE:  TO GET THE ANGLE ASSOCIATED WITH A GRAPHIC TEXT           *
' *                                                                      *
' *  GIVEN:     aGraphicText = graphic text to be processed              *
' *                                                                      *
' *  RETURN:    aAngle        = graphic text angle (degrees)             *
' *                                                                    *
' *  Dim aGraphicText As IElement                                      *
' *  Dim avGraphicTextGetAngle As Double                               *
' *                                                                    *
Public  Sub  avGraphicTextGetStyle(aGraphicText  As  IElement,  _
                 aFont, aSize, aBold, aItalic, aColor As iColor)
' *                                                                    *
' *   PURPOSE:   TO GET THE TEXT STYLE ATTRIBUTES THAT ARE ASSOCIATED  *
' *                WITH A GRAPHIC TEXT, THIS INCLUDES THE ATTRIBUTES FOR  *
' *              FONT, SIZE, BOLD, ITALIC AND COLOR                    *
' *                                                                    *
' *   GIVEN:     aGraphicText = graphic text to be processed           *
' *                                                                    *
' *  RETURN:    aFont         = font name                              *
' *              aSize         = font size                             *
' *               aBold         = font style (1 = normal, 2 = bold)    *
' *               aItalic       = font style (1 = normal, 2 = italic)  *
' *              aColor        = font color (color object)             *
' *                                                                    *
' *  Dim aGraphicText As IElement                                      *
' *  Dim aFont As String, aSize As Double                              *
' *   Dim aBold As Integer, aItalic As Integer, aColor As IColor       *
' *                                                                    *
Public  Function  avGraphicTextGetSymbol(aGraphicText  _
                                        As IElement) As ISymbol
' *                                                                    *
' *   PURPOSE:  TO GET THE SYMBOL ASSOCIATED WITH A GRAPHIC TEXT       *
' *                                                                    *
' *   GIVEN:     aGraphicText             = graphic text to be processed  *
' *                                                                    *
' *   RETURN:     avGraphicTextGetSymbol  = symbol describing the graphic *
' *                                           text, its properties such as  *
' *                                           font, size, bold, italic and  *
' *                                         color                      *
' *                                                                    *
' *  Dim aGraphicText As IElement                                      *
' *  Dim avGraphicTextGetSymbol As ISymbol                             *
' *                                                                    *
Public  Function  avGraphicTextGetText(aGraphicText  As  IElement)
' *                                                                    *
' *   PURPOSE:   TO GET THE TEXT STRING ASSOCIATED WITH A GRAPHIC TEXT *
' *                                                                    *
' *   GIVEN:     aGraphicText            = graphic text to be processed   *
' *                                                                    *
' *   RETURN:    avGraphicTextGetAngle = the text string               *
' *                                                                    *
' *  Dim aGraphicText As IElement                                      *
' *  Dim avGraphicTextGetText As String                                *
' *                                                                    *
Public  Function  avGraphicTextMake(aString,  _
                                    theGeom As IGeometry) As IElement
' *                                                                    *
' *   PURPOSE:   TO CREATE A GRAPHIC TEXT THAT CAN BE ADDED TO THE     *
' *              GRAPHICS LAYER                                        *
' *                                                                    *
' *   GIVEN:     aString             = text string that is associated  *
' *                                    with the graphic text           *
' *               theGeom             = geometry describing the graphic  *
' *                                                                    *
' *   RETURN:    avGraphicTextMake = the graphic that can be added to  *
' *                                    the graphics layer              *
' *                                                                    *
' *   Dim aString As String, theGeom As IGeometry                      *
' *  Dim avGraphicTextMake As IElement                                 *
' *                                                                    *
Public  Sub  avGraphicTextSetAngle(aGraphicText  As  IElement,  aAngle)
' *                                                                    *
' *   PURPOSE:   TO SET THE ANGLE ASSOCIATED WITH A GRAPHIC TEXT       *
' *                                                                    *
' *   GIVEN:     aGraphicText = graphic text to be processed           *
' *               aAngle        = angle to be assigned (degrees)       *
' *                                                                    *
' *  RETURN:   nothing                                                 *
' *                                                                    *
' *  Dim aGraphicText As IElement                                      *
' *  Dim aAngle As Double                                              *
' *                                                                    *
Public  Sub  avGraphicTextSetStyle(aGraphicText  As  IElement,  _
                 aFont, aSize, aBold, aItalic, aColor As iColor)
' *                                                                    *
' *   PURPOSE:  TO SET THE TEXT STYLE ATTRIBUTES THAT ARE ASSOCIATED   *
' *               WITH A GRAPHIC TEXT, THIS INCLUDES THE ATTRIBUTES FOR  *
' *              FONT, SIZE, BOLD, ITALIC AND COLOR                    *
' *                                                                    *
' *   GIVEN:     aGraphicText = graphic text to be processed           *
' *              aFont         = font name                             *
' *              aSize         = font size                             *
' *               aBold         = font style (1 = normal, 2 = bold)    *
' *               aItalic       = font style (1 = normal, 2 = italic)  *
' *              aColor        = font color (color object)             *
' *                                                                    *
' *  RETURN:   nothing                                                 *
' *                                                                    *
' *  Dim aGraphicText As IElement                                      *
' *  Dim aFont As String, aSize As Double                              *
' *   Dim aBold As Integer, aItalic As Integer, aColor As IColor       *
' *                                                                    *
Public  Sub  avGraphicTextSetSymbol(aGraphicText  As  IElement,  _
                                   pTextSymbol As ISymbol)
' *                                                                    *
' *   PURPOSE:  TO SET THE SYMBOL ASSOCIATED WITH A GRAPHIC TEXT       *
' *                                                                    *
' *   GIVEN:     aGraphicText = graphic text to be processed           *
' *                pTextSymbol   = symbol describing the graphic text, its *
' *                                properties such as font, size, bold,  *
' *                               italic and color                     *
' *                                                                    *
' *  RETURN:   nothing                                                 *
' *                                                                    *
' *   Dim aGraphicText As IElement, pTextSymbol As ISymbol             *
' *                                                                    *
Public  Sub  avGraphicTextSetText(aGraphicText  As  IElement,  aString)
' *                                                                    *
' *   PURPOSE:  TO SET THE TEXT STRING ASSOCIATED WITH A GRAPHIC TEXT  *
' *                                                                    *
' *   GIVEN:     aGraphicText = graphic text to be processed           *
' *               aString       = the text string to be assigned       *
' *                                                                    *
' *  RETURN:   nothing                                                 *
' *                                                                    *
' *   Dim aGraphicText As IElement, aString As String                  *
' *                                                                    *
Public  Function  avHasM(aGeom  As  IGeometry)
' *                                                                    *
' *   PURPOSE:  TO CHECK IF A GEOMETRY OBJECT HAS AN M ATTRIBUTE       *
' *                                                                    *
' *   GIVEN:     aGeom  = geometry object to be processed              *
' *                                                                    *
' *   RETURN:    avHasM = flag denoting if the given geometry has an M *
' *                        true = it does, false = it does not         *
' *                                                                    *
' *  Dim aGeom As IGeometry                                            *
' *  Dim avHasM As Boolean                                             *
' *                                                                    *
Public  Function  avHasZ(aGeom  As  IGeometry)
' *                                                                    *
' *   PURPOSE:  TO CHECK IF A GEOMETRY OBJECT HAS A Z ATTRIBUTE        *
' *                                                                    *
' *   GIVEN:     aGeom   = geometry object to be processed              *
' *                                                                     *
' *   RETURN:    avHasZ  = flag denoting if the given geometry has a Z  *
' *                        true = it does, false = it does not          *
' *                                                                     *
' *  Dim aGeom As IGeometry                                             *
' *  Dim avHasZ As Boolean                                              *
' *                                                                     *
Public  Sub  avInit(m_pApp  As  IApplication)
' *                                                                     *
' *   PURPOSE:   TO INITIALIZE THE GLOBAL VARIABLES THAT ARE USED BY    *
' *              THE AVENUE WRAPS                                       *
' *                                                                     *
' *   GIVEN:     m_pApp  = the IApplication object                      *
' *                                                                     *
' *   RETURN:    nothing                                                *
' *                                                                     *
' *   NOTE:       This routine needs to be called only once, typically  *
' *               when the project (*.mxd) is opened or when the        *
' *               extension is initially loaded.  When the Avenue Wraps *
' *               are used in a project file (*.mxd) a call to avInit   *
' *               can be made in the OpenDocument event for the         *
' *               procedure MxDocument. In so doing when the project is *
' *               opened, avInit will be called to initialize the global *
' *               variables referenced in avInit                        *
' *                                                                     *
' *  Dim m_pApp As IApplication                                         *
' *                                                                     *
Public  Function  avIntersects(aShape1  As  IGeometry,  _
                               aShape2 As IGeometry) As Boolean
' *                                                                     *
' *   PURPOSE:   TO CHECK IF TWO SHAPES INTERSECT EACH OTHER            *
' *                                                                     *
' *   GIVEN:     aShape1      = base shape                              *
' *                aShape2      = second shape to be intersected with the *
' *                             base shape                              *
' *                                                                     *
' *   RETURN:    avIntersects  = intersection state of the input objects *
' *                             true = intersect, false = do not        *
' *                                                                     *
' *  Dim aShape1 As IGeometry, aShape2 As IGeometry                     *
' *  Dim avIntersects As Boolean                                        *
' *                                                                     *
Public  Sub  avInterval(pmxDoc  As  IMxDocument,  theTheme,  aField,  numClass)
' *                                                                     *
' *   PURPOSE:   TO SET THE LEGEND THAT IS ASSOCIATED WITH A THEME      *
' *                TO BE OF QUANTILE TYPE WITH THE CLASSES DETERMINED BY *
' *              USING AN EQUAL INTERVAL METHOD                         *
' *                                                                     *
' *   GIVEN:     pmxDoc     = the active view                           *
' *              theTheme   = theme to be processed                     *
' *               aField     = field name that theme is to be classified *
' *                            upon                                     *
' *               numClass   = number of classes to be generated        *
' *                                                                     *
' *   RETURN:    nothing                                                *
' *                                                                     *
' *   NOTE:       (a) Divides the features in the theme into numClass   *
' *                   classifications of equal size using the values in *
' *                   aField. This is only supported for numeric fields *
' *               (b) After a theme has been classified you must use    *
' *                   avGetLegend to get the new legend that reflects   *
' *                   the new classification if you wish to manipulate  *
' *                   the labels or symbols in the classification       *
' *                                                                     *
' *  Dim pmxDoc As IMxDocument                                          *
' *  Dim theTheme As Variant, aField As String                          *
' *  Dim numClass As Long                                               *
' *                                                                     *
```

```
Public  Sub  avInvalidateTOC(name)
' *                                                                  *
' *  PURPOSE:  REFRESH THE TABLE OF CONTENTS                         *
' *                                                                  *
' *  GIVEN:     name     = name of theme or table in the Table of    *
' *                          Contents to be refreshed, if NULL the entire *
' *                          Table of Contents will be refreshed     *
' *                                                                  *
' *  RETURN:   nothing                                               *
' *                                                                  *
' *  Dim name As Variant                                             *
' *                                                                  *
Public  Sub  avInvSelFeatures(pmxDoc  As  IMxDocument,  selThmList)
' *                                                                  *
' *   PURPOSE:   TO REDRAW THE SELECTED FEATURES FOR A SET OF THEMES *
' *                                                                  *
' *  GIVEN:     pmxDoc      = the active view                        *
' *               selThmList = list of themes with selected features *
' *                                                                  *
' *  RETURN:   nothing                                               *
' *                                                                  *
' *   NOTE:       Structure of selThmList is a sequential list with two *
' *               items per theme, name of the theme and the number of *
' *              selected features in the theme, so that:            *
' *             Item 1: name of theme 1                              *
' *              Item 2: number of selected features in theme 1      *
' *             Item 3: name of theme 2                              *
' *              Item 4: number of selected features in theme 2      *
' *             Repeat Items 1 and 2 for each theme                  *
' *                                                                  *
' *  Dim pmxDoc As IMxDocument                                       *
' *  Dim selThmList As New Collection                                *
' *                                                                  *
Public  Function  avIsCoverage(name)  As  Boolean
' *                                                                  *
' *  PURPOSE:  DETERMINE IF A LAYER IS A COVERAGE                    *
' *                                                                  *
' *  GIVEN:     name     = name of input object to be checked        *
' *                                                                  *
' *   RETURN:    avIsCoverage  = flag denoting whether the input object *
' *                              is a coverage or not                *
' *                              true = is, false = not              *
' *                                                                  *
' *  Dim name As Variant                                             *
' *  Dim avIsCoverage As Boolean                                     *
' *                                                                  *
Public  Function  avIsEditable(name)  As  Boolean
' *                                                                  *
' *   PURPOSE:   DETERMINE IF A LAYER OR TABLE IS EDITABLE OR NOT    *
' *                                                                  *
' *  GIVEN:     name           = name of theme or table for which its *
' *                                editability status is to be checked *
' *                                                                  *
' *   RETURN:    avIsEditable  = editability state of the layer or table *
' *                                true = editable, false = not editable *
' *                                                                  *
' *  Dim name As Variant                                             *
' *  Dim avIsEditable As Boolean                                     *
' *                                                                  *
Public  Function  avIsFTheme(name)  As  Boolean
' *                                                                  *
' *   PURPOSE:   DETERMINE IF A LAYER IS OF FEATURE LAYER TYPE OR NOT *
' *                                                                  *
' *  GIVEN:     name        = name of input object for which its     *
' *                            feature layer type is to be checked   *
' *                                                                  *
' *   RETURN:    avIsFTheme  = flag denoting whether the input object *
' *                            is a feature layer or not             *
' *                            true = is, false = not                *
' *                                                                           *
' *  Dim name As Variant                                                      *
' *  Dim avIsFTheme As Boolean                                                *
' *                                                                           *
Public  Function  avIsJoined(aVTab,  updSelFeat)  As  Boolean
' *                                                                           *
' *   PURPOSE:   TO DETERMINE IF A VTAB HAS A JOIN (RELATE) OR NOT AND        *
' *                IF SO UPDATE THE SELECTED FEATURES REFLECTING THE JOIN     *
' *                                                                           *
' *  GIVEN:     aVTab       = name of VTab to be processed                    *
' *               updSelFeat = flag denoting whether or not the selected      *
' *                            features associated with the join should       *
' *                            be updated (redrawn/reselected)                *
' *                            true = redraw, false = do not redraw           *
' *                                                                           *
' *   RETURN:    avIsJoined  = flag denoting whether the input object         *
' *                            has a join or not                              *
' *                            true = has join, false = not joined            *
' *                                                                           *
' *  Dim aVTab As String, updSelFeat As Boolean                               *
' *  Dim avIsJoined As Boolean                                                *
' *                                                                           *
Public  Function  avIsLinked(aVTab)  As  Boolean
' *                                                                           *
' *   PURPOSE:   TO DETERMINE IF A VTAB HAS LINKS (RELATES) OR NOT            *
' *                                                                           *
' *  GIVEN:     aVTab       = name of VTab to be processed                    *
' *                                                                           *
' *   RETURN:    avIsLinked  = flag denoting whether the input object         *
' *                            has links or not                               *
' *                            true = has links, false = not linked           *
' *                                                                           *
' *  Dim aVTab As String                                                      *
' *  Dim avIsLinked As Boolean                                                *
' *                                                                           *
Public  Function  avIsSDE(name)  As  Boolean
' *                                                                           *
' *   PURPOSE:   DETERMINE IF A LAYER IS AN SDE GEODATABASE                   *
' *                                                                           *
' *  GIVEN:     name     = name of input object to be checked                 *
' *                                                                           *
' *   RETURN:    avIsSDE  = flag denoting whether the input object is         *
' *                         an SDE geodatabase or not                         *
' *                         true = is, false = not                            *
' *                                                                           *
' *  Dim name As Variant                                                      *
' *  Dim avIsSDE As Boolean                                                   *
' *                                                                           *
Public  Function  avIsVisible(name)  As  Boolean
' *                                                                           *
' *   PURPOSE:   DETERMINE IF AN OBJECT IS VISIBLE OR NOT                     *
' *                                                                           *
' *  GIVEN:     name          = name of input object for which its            *
' *                             visibility status is to be checked            *
' *                                                                           *
' *   RETURN:    avIsVisible = visible state of the input object              *
' *                             true = visible, false = not visible           *
' *                                                                           *
' *  Dim name As Variant                                                      *
' *  Dim avIsVisible As Boolean                                               *
' *                                                                           *
Public  Function  avIsWithin(aShape1  As  IGeometry,  _
                              aShape2 As IGeometry, _
                              aDist) As Boolean
' *                                                                           *
' *   PURPOSE:   TO DETERMINE IF A SHAPE IS WITHIN A DISTANCE OF              *
' *              ANOTHER SHAPE                                                *
' *                                                                           *
' *  GIVEN:     aShape1     = geometry object to be checked                   *
' *              aShape2     = source geometry object aShape1 is to be    *
' *                            compared against                           *
' *             aDist       = distance value                              *
' *                                                                       *
' *   RETURN:    avIsWithin  = flag denoting if aShape1 is within a user  *
' *                             specified distance of aShape2             *
' *                             true = it is, false = it is not           *
' *                                                                       *
' *  Dim aShape1 As IGeometry, aShape2 As IGeometry                       *
' *  Dim aDist As Double                                                  *
' *  Dim avIsWithin As Boolean                                            *
' *                                                                       *
Public  Function  avJoin(aVTab1,  aField1,  aVTab2,  aField2)  As  Integer
' *                                                                       *
' *   PURPOSE:   TO JOIN aVTab2 to aVTab1 USING USER SPECIFIED FIELD      *
' *              NAMES                                                    *
' *                                                                       *
' *   GIVEN:      aVTab1   = name of VTab which aVTab2 is to be joined to *
' *               aField1 = field in aVTab1 join is based upon            *
' *               aVTab2   = name of VTab to be joined to aVTab1          *
' *               aField2 = field in aVTab2 join is based upon            *
' *                                                                       *
' *  RETURN:    avJoin   = error flag                                     *
' *                        0 : no error                                   *
' *                        1 : error detected                             *
' *                        2 : aVTab1 does not exist                      *
' *                        3 : aVTab2 does not exist                      *
' *                                                                       *
' *  Dim aVTab1 As String, aField1 As String                              *
' *  Dim aVTab2 As String, aField2 As String                              *
' *  Dim avJoin As Integer                                                *
' *                                                                       *
Public  Sub  avLegendGetSymbols(thelegend  As  IFeatureRenderer,  symbList)
' *                                                                       *
' *   PURPOSE:   GET A LIST OF SYMBOLS APPEARING IN A LEGEND              *
' *                                                                       *
' *   GIVEN:     theLegend = legend to be processed                       *
' *                                                                       *
' *   RETURN:    symbList  = list of symbols used in the legend           *
' *                                                                       *
' *  Dim theLegend As IFeatureRenderer                                    *
' *  Dim symbList As New Collection                                       *
' *                                                                       *
Public  Sub  avLegendSetSymbols(thelegend  As  IFeatureRenderer,  symbList)
' *                                                                       *
' *   PURPOSE:   TO SET THE SYMBOLS USED IN THE CLASSIFICATIONS WHICH     *
' *              APPEAR IN A LEGEND                                       *
' *                                                                       *
' *   GIVEN:     theLegend = legend to be processed                       *
' *                symbList  = list of symbols that are to be assigned    *
' *                            to the classifications in a legend         *
' *                                                                       *
' *  RETURN:    nothing                                                   *
' *                                                                       *
' *  Dim theLegend As IFeatureRenderer                                    *
' *  Dim symbList As New Collection                                       *
' *                                                                       *
Public  Sub  avLineFileClose(aFile)
' *                                                                       *
' *  PURPOSE:  CLOSE A FILE CONNECTION                                    *
' *                                                                       *
' *   GIVEN:     aFile    = textstream object to be processed             *
' *                                                                       *
' *  RETURN:    nothing                                                   *
' *                                                                       *
' *  Dim aFile                                                            *
' *                                                                       *
```

```
Public  Function  avLineFileMake(aFileName,  aFilePerm)
' *                                                                          *
' *   PURPOSE:   OPEN A FILE CONNECTION FOR READING AND/OR WRITING           *
' *                                                                          *
' *   GIVEN:     aFileName       = the name of the file to be processed      *
' *               aFilePerm       = the type of file connection desired      *
' *                                  READ   : open file for reading          *
' *                                  WRITE  : open file for writing          *
' *                                  APPEND : open file for appending        *
' *                                                                          *
' *   RETURN:    avLineFileMake = textstream object that can be used         *
' *                                  for reading and/or writing operations   *
' *                                                                          *
' *   NOTE:      If an error is detected in opening the file no error        *
' *               message is generated, however, avLineFileMake will be      *
' *               set to NOTHING so that the calling routine will need       *
' *              to perform the appropriate error checking                   *
' *                                                                          *
' *   Dim aFileName As String, aFilePerm As String                           *
' *   Dim avLineFileMake                                                     *
' *                                                                          *
Public  Function  avLink(aVTab1,  aField1,  aVTab2,  aField2)  As  Integer
' *                                                                          *
' *   PURPOSE:   TO LINK (RELATE) aVTab2 to aVTab1 USING USER SPECIFIED      *
' *              FIELD NAMES                                                 *
' *                                                                          *
' *   GIVEN:     aVTab1   = name of VTab which aVTab2 is to be linked to     *
' *              aField1 = field in aVTab1 link is based upon                *
' *              aVTab2   = name of VTab to be linked to aVTab1              *
' *              aField2 = field in aVTab2 link is based upon                *
' *                                                                          *
' *   RETURN:    avLink   = error flag                                       *
' *                          0 : no error                                    *
' *                          1 : error detected                              *
' *                          2 : aVTab1 does not exist                       *
' *                          3 : aVTab2 does not exist                       *
' *                                                                          *
' *   Dim aVTab1 As String, aField1 As String                                *
' *   Dim aVTab2 As String, aField2 As String                                *
' *   Dim avLink As Integer                                                  *
' *                                                                          *
Public  Sub  avListFiles(aDIR,  filType,  filList)
' *                                                                          *
' *   PURPOSE:   GET A LIST OF FILES IN A DIRECTORY                          *
' *                                                                          *
' *   GIVEN:     aDir     = the directory to be scanned                      *
' *              filType = the type of files to be searched for              *
' *                          vbNormal     0 Specifies files with no          *
' *                                          attributes                      *
' *                          vbReadOnly   1 Specifies read-only files in     *
' *                                          addition to files with no       *
' *                                          attributes                      *
' *                          vbHidden     2 Specifies hidden files in        *
' *                                          addition to files with no       *
' *                                          attributes                      *
' *                          VbSystem     4 Specifies system files in        *
' *                                          addition to files with no       *
' *                                          attributes                      *
' *                          vbVolume     8 Specifies volume label; if       *
' *                                          any other attribute is given    *
' *                                          vbVolume is ignored             *
' *                          vbDirectory 16 Specifies directories or         *
' *                                          folders in addition to files    *
' *                                          with no attributes              *
' *                                                                          *
' *   RETURN:    filList = list of files that were found as specified        *
' *                          by the filType argument                         *
' *                                                                          *
' *   NOTE:      The filType argument can be specified either as the         *
' *               numeric value, shown above, or as the VB keyword that   *
' *             is shown                                                  *
' *                                                                       *
' *  Dim aDir As String                                                   *
' *  Dim filType As Integer                                               *
' *  Dim filList As New Collection                                        *
' *                                                                       *
Public Sub avMoveTo(aDoc As IUnknown, aLeft, aTop)
' *                                                                       *
' *  PURPOSE:  TO REPOSITION A WINDOW OBJECT                              *
' *                                                                       *
' *  GIVEN:    aDoc    = the window object                                *
' *              aLeft   = distance from the left side of the screen      *
' *             aTop    = distance from the top of the screen             *
' *                                                                       *
' *  RETURN:   nothing                                                    *
' *                                                                       *
' *  Dim aDoc As IUnknown                                                 *
' *  Dim aLeft, aTop As Long                                              *
' *                                                                       *
Public Sub avMsgBox(aMessage, Optional aButtons, Optional Heading)
' *                                                                       *
' *  PURPOSE:  DISPLAY A MESSAGE BOX SIMILAR TO THE MESSAGE BOX THAT      *
' *              IS DISPLAYED BY THE VB MSGBOX FUNCTION BUT IN A          *
' *              POSITION CONTROLLED BY GLOBAL VARIABLES RATHER THAN IN   *
' *             THE CENTER OF THE APPLICATION WINDOW                      *
' *                                                                       *
' *  GIVEN:    aMessage = the message to be displayed                     *
' *              aButtons = Optional, numeric expression that is the      *
' *                         sum of values specifying the number and       *
' *                         type of buttons to display, the icon style    *
' *                         to use, the identity of the default button,   *
' *                         and the modality of the message box. If       *
' *                         omitted, the default value for buttons is 0   *
' *              Heading  = Optional, string expression displayed in      *
' *                         the title bar of the dialog box. If omitted   *
' *                        the application name is used                   *
' *                                                                       *
' *  RETURN:   nothing                                                    *
' *                                                                       *
' *  Dim aMessage As Variant, aButtons As Variant, Heading As Variant     *
' *                                                                       *
Public Sub avMsgBoxChoice(aList, aMsg, Heading, ians)
' *                                                                       *
' *  PURPOSE:  DISPLAY A CHOICE MESSAGE BOX WHICH CONTAINS A LIST OF      *
' *             STRING ITEMS                                              *
' *                                                                       *
' *  GIVEN:    aList   = the list of items to be displayed                *
' *             aMsg    = the message to be displayed                     *
' *             Heading = message box caption                             *
' *                                                                       *
' *  RETURN:   ians     = the item selected by the user, if the user      *
' *                        Cancels the command: ians will be equal to     *
' *                       NULL                                            *
' *                                                                       *
' *  Dim aList As New Collection, aMsg As Variant, Heading As Variant     *
' *  Dim ians As Variant                                                  *
' *                                                                       *
Public Sub avMsgBoxChoice2(aList, aMsg, Heading, itemList)
' *                                                                       *
' *  PURPOSE:  DISPLAY A LIST OF STRINGS WITHIN A MESSAGE BOX             *
' *                                                                       *
' *  GIVEN:    aList    = the list of strings to be displayed             *
' *             aMsg     = the message to be displayed                    *
' *             Heading  = message box caption                            *
' *                                                                       *
' *  RETURN:   itemList = the list of items selected by the user,         *
' *                         will be an empty list if the user cancels     *
' *                        the operation                                  *
' *                                                                    *
' *   NOTE:       This subroutine is the same as avMsgBoxChoice with the *
' *               exception that once the user makes a selection the   *
' *               message box is terminated. In addition, the OK and   *
' *               Cancel buttons are not displayed on the message box  *
' *                                                                    *
' *   Dim aList As New Collection, aMsg As Variant, Heading As Variant *
' *   Dim itemList As New Collection                                   *
' *                                                                    *
Public Function avMsgBoxF(aMessage, _
                           Optional aButtons, Optional Heading) As Integer
' *                                                                    *
' *   PURPOSE:   DISPLAY A MESSAGE BOX SIMILAR TO THE MESSAGE BOX THAT  *
' *              IS DISPLAYED BY THE VB MSGBOX FUNCTION BUT IN A       *
' *              POSITION CONTROLLED BY GLOBAL VARIABLES RATHER THAN IN *
' *              THE CENTER OF THE APPLICATION WINDOW                  *
' *                                                                    *
' *   GIVEN:     aMessage  = the message to be displayed               *
' *              aButtons  = Optional, numeric expression that is the  *
' *                          sum of values specifying the number and   *
' *                          type of buttons to display, the icon style *
' *                          to use, the identity of the default button *
' *                          and the modality of the message box. If   *
' *                          omitted, the default value used is 0      *
' *                          Available button values:                  *
' *                            0 : (vbOKOnly) OK                       *
' *                            1 : (vbOKCancel) OK and Cancel          *
' *                            3 : (vbYesNoCancel) Yes, No, and Cancel *
' *                            4 : (vbYesNo) Yes and No                *
' *                          Available icon values:                    *
' *                            32 : (vbQuestion) Warning Query         *
' *                            48 : (vbExclamation) Warning Message    *
' *                            64 : (vbInformation) Information Message *
' *                          Available default button values:          *
' *                             0 : (vbDefaultButton1) Button 1 default *
' *                           256 : (vbDefaultButton2) Button 2 default *
' *              Heading   = Optional, string expression displayed in  *
' *                          the title bar of the dialog box. If not   *
' *                          specified the application name is used    *
' *                                                                    *
' *   RETURN:    avMsgBoxF = indicates which button the user clicked   *
' *                          1 : OK        (vbOK)                      *
' *                          2 : Cancel    (vbCancel)                  *
' *                          3 : Abort     (vbAbort)                   *
' *                          4 : Retry     (vbRetry)                   *
' *                          5 : Ignore    (vbIgnore)                  *
' *                          6 : Yes       (vbYes)                     *
' *                          7 : No        (vbNo)                      *
' *                                                                    *
' *   Dim aMessage As Variant, aButtons As Variant, Heading As Variant *
' *   Dim avMsgBoxF As Integer                                         *
' *                                                                    *
Public Sub avMsgBoxInfo(aMessage, heading)
' *                                                                    *
' *   PURPOSE:   DISPLAY AN INFORMATION MESSAGE BOX                    *
' *                                                                    *
' *   GIVEN:     aMessage = the message to be displayed                *
' *              Heading  = message box caption                        *
' *                                                                    *
' *   RETURN:    nothing                                               *
' *                                                                    *
' *   Dim aMessage As Variant, Heading As Variant                      *
' *                                                                    *
Public Sub avMsgBoxInput(aMsg, heading, aDefault, ians)
' *                                                                    *
' *   PURPOSE:   DISPLAY A SINGLE LINE INPUT MESSAGE BOX               *
' *                                                                    *
' *   GIVEN:     aMsg     = the message to be displayed                *
' *              Heading  = message box caption                        *
' *              aDefault = the default button value                    *
' *                                                                     *
' *   RETURN:    ians      = the response from the user, if the user    *
' *                            Cancels the command: ians will be equal to *
' *                          NULL                                       *
' *                                                                     *
' *   Dim aMsg As Variant, Heading As Variant, aDefault As Variant      *
' *  Dim ians As Variant                                                *
' *                                                                     *
Public  Sub  avMsgBoxList(aList,  aMsg,  Heading,  ians)
' *                                                                     *
' *   PURPOSE:   DISPLAY A LIST OF STRINGS WITHIN A MESSAGE BOX         *
' *                                                                     *
' *   GIVEN:     aList    = the list of strings to be displayed         *
' *              aMsg     = the message to be displayed                 *
' *              Heading = message box caption                          *
' *                                                                     *
' *   RETURN:    ians      = the button that was selected, ians will be *
' *                          equal to vbOK or vbCancel                  *
' *                                                                     *
' *   Dim aList As New Collection, aMsg As Variant, Heading As Variant *
' *  Dim ians As Integer                                                *
' *                                                                     *
Public  Sub  avMsgBoxMultiInput(aMsg,  heading,  labels,  defaults,  aList)
' *                                                                     *
' *   PURPOSE:   DISPLAY A MULTI-INPUT LINE MESSAGE BOX                 *
' *                                                                     *
' *   GIVEN:     aMsg      = the message to be displayed                *
' *              Heading  = message box caption                         *
' *                labels    = list of labels for each of the items that *
' *                          are prompted for                           *
' *                defaults = list of default values for each of the    *
' *                           items that are prompted for               *
' *                                                                     *
' *   RETURN:    aList     = list of responses for each of the items    *
' *                            that were displayed, if the user Cancels *
' *                            the command: aList will be an empty list, *
' *                           that is, aList.Count will equal 0         *
' *                                                                     *
' *   NOTE:        There is no limit to the number of items that can be *
' *                  prompted for, the difficulty however will be that the *
' *                  dialog box may exceed the visible area of the screen. *
' *                  Recommend using avMsgBoxMultiInput2 when more than 12 *
' *              to 15 items are to be displayed.                       *
' *                                                                     *
' *   Dim aMsg As Variant, Heading As Variant                           *
' *   Dim labels As New Collection, defaults As New Collection          *
' *  Dim aList As New Collection                                        *
' *                                                                     *
Public  Sub  avMsgBoxMultiInput2(aMsg,  heading,  labels,  defaults,  aList)
' *                                                                     *
' *   PURPOSE:   DISPLAY A MULTI-INPUT LINE MESSAGE BOX WITH A BACK     *
' *                BUTTON WHEN MORE THAN 10 ITEMS ARE TO BE DISPLAYED   *
' *                                                                     *
' *   GIVEN:     aMsg      = the message to be displayed                *
' *              Heading  = message box caption                         *
' *                labels    = list of labels for each of the items that *
' *                          are prompted for                           *
' *                defaults = list of default values for each of the    *
' *                           items that are prompted for               *
' *                                                                     *
' *   RETURN:    aList     = list of responses for each of the items    *
' *                            that were displayed, if the user Cancels *
' *                            the command: aList will be an empty list, *
' *                           that is, aList.Count will equal 0         *
' *                                                                     *
' *   Dim aMsg As Variant, Heading As Variant                           *
' *   Dim labels As New Collection, defaults As New Collection          *
' *  Dim aList As New Collection                                        *
' *                                                                   *
Public  Sub  avMsgBoxMultiList(aList,  aMsg,  Heading,  itemList)
' *                                                                   *
' *   PURPOSE:   DISPLAY A LIST OF STRINGS WITHIN A MESSAGE BOX WITH  *
' *              THE ABILITY TO SELECT MULTIPLE ITEMS                  *
' *                                                                   *
' *   GIVEN:     aList     = the list of strings to be displayed      *
' *              aMsg      = the message to be displayed              *
' *              Heading   = message box caption                      *
' *                                                                   *
' *   RETURN:    itemList = the list of items selected by the user,  *
' *                          will be an empty list if the user cancels *
' *                          the operation                             *
' *                                                                   *
' *   Dim aList As New Collection, aMsg As Variant, Heading As Variant *
' *   Dim itemList As New Collection                                   *
' *                                                                   *
Public  Sub  avMsgBoxWarning(aMessage,  Heading)
' *                                                                   *
' *   PURPOSE:   DISPLAY A WARNING MESSAGE BOX                         *
' *                                                                   *
' *   GIVEN:     aMessage = the message to be displayed               *
' *              Heading  = message box caption                       *
' *                                                                   *
' *   RETURN:    nothing                                               *
' *                                                                   *
' *   Dim aMessage As Variant, Heading As Variant                      *
' *                                                                   *
Public  Sub  avMsgBoxYesNo(aMessage,  Heading,  aDefault,  ians)
' *                                                                   *
' *   PURPOSE:   DISPLAY A YES, NO MESSAGE BOX                         *
' *                                                                   *
' *   GIVEN:     aMessage = the message to be displayed               *
' *              Heading  = message box caption                       *
' *              aDefault = the default button setting                *
' *                         true = yes, false = no                     *
' *                                                                   *
' *   RETURN:    ians      = the button that was selected, ians will be *
' *                          equal to vbYes or vbNo                    *
' *                                                                   *
' *   Dim aMessage As Variant, Heading As Variant, aDefault As Boolean *
' *   Dim ians As Integer                                              *
' *                                                                   *
Public  Sub  avMsgBoxYesNoCancel(aMessage,  heading,  aDefault,  ians)
' *                                                                   *
' *   PURPOSE:   DISPLAY A YES, NO, CANCEL MESSAGE BOX                 *
' *                                                                   *
' *   GIVEN:     aMessage = the message to be displayed               *
' *              Heading  = message box caption                       *
' *              aDefault = the default button setting                *
' *                         true = yes, false = no                     *
' *                                                                   *
' *   RETURN:    ians      = the button that was selected, ians will be *
' *                          equal to vbYes, vbNo or vbCancel          *
' *                                                                   *
' *   Dim aMessage As Variant, Heading As Variant, aDefault As Boolean *
' *   Dim ians As Integer                                              *
' *                                                                   *
Public  Function  avMultipointMake(aPntList)  As  IMultipoint
' *                                                                   *
' *   PURPOSE:   TO CREATE A MULTIPOINT OBJECT FROM A LIST OF POINTS   *
' *                                                                   *
' *   GIVEN:     aPntList          = list of point objects            *
' *                                                                   *
' *   RETURN:    avMultipointMake = the multipoint feature            *
' *                                                                   *
' *   Dim aPntList As New Collection                                   *
' *   Dim avMultipointMake As IMultipoint                              *
' *                                                                   *
```

```
Public  Sub  avNatural(pmxDoc  As  IMxDocument,  theTheme,  aField,  numClass)
' *                                                                         *
' *   PURPOSE:   TO SET THE LEGEND THAT IS ASSOCIATED WITH A THEME          *
' *                TO BE OF QUANTILE TYPE WITH THE CLASSES DETERMINED BY    *
' *              USING THE NATURAL BREAK METHOD                             *
' *                                                                         *
' *  GIVEN:     pmxDoc     = the active view                                *
' *              theTheme  = theme to be processed                          *
' *               aField     = field name that theme is to be classified    *
' *                            upon                                         *
' *               numClass   = number of classes to be generated            *
' *                                                                         *
' *  RETURN:    nothing                                                     *
' *                                                                         *
' *   NOTE:      Divides the features in the theme into numClass            *
' *               classifications of equal size using the values in         *
' *               aField. This is only supported for numeric fields         *
' *                                                                         *
' *  Dim pmxDoc As IMxDocument                                              *
' *  Dim theTheme As Variant, aField As String                              *
' *  Dim numClass As Long                                                   *
' *                                                                         *
Public  Sub  avObjGetName(theObject  As  IUnknown,  aName)
' *                                                                         *
' *  PURPOSE:  TO GET THE NAME OF AN OBJECT                                 *
' *                                                                         *
' *  GIVEN:     theObject = object to be processed                          *
' *                                                                         *
' *  RETURN:    aName      = name assigned to the object                    *
' *                                                                         *
' *   NOTE:      This subroutine checks if the object matches a known       *
' *               type, if so, the appropriate interface is selected        *
' *               and the given name assigned to the object is extracted    *
' *              If a match is not made, the name will be NULL              *
' *                                                                         *
' *  Dim theObject As IUnknown                                              *
' *  Dim aName As Variant                                                   *
' *                                                                         *
Public  Sub  avObjSetName(theObject  As  IUnknown,  aName)
' *                                                                         *
' *  PURPOSE:  TO SET THE NAME FOR AN OBJECT                                *
' *                                                                         *
' *  GIVEN:     theObject = object which is to be named                     *
' *              aName      = name to be assigned to the object             *
' *                                                                         *
' *  RETURN:    nothing                                                     *
' *                                                                         *
' *   NOTE:      This subroutine checks if the object matches a known       *
' *               type, if so, the appropriate interface is selected        *
' *               and the given name assigned to the object. If a match     *
' *              is not made, the object is left unaltered                  *
' *                                                                         *
' *  Dim theObject As IUnknown                                              *
' *  Dim aName As String                                                    *
' *                                                                         *
Public  Function  avOpenFeatClass(opmode,  sDir,  _
                                    sName, aFCtype) As IUnknown
' *                                                                         *
' *   PURPOSE:   OPEN A DATASET OR A FEATURECLASS IN A DATASET FOR          *
' *              PROCESSING                                                 *
' *                                                                         *
' *  GIVEN:     opmode           = type of dataset to be opened             *
' *                                1 : shapefile                            *
' *                                2 : raster                               *
' *                                3 : tin                                  *
' *                                4 : coverage                             *
' *                                  5 : access database feature class      *
' *                                  6 : access database feature dataset    *
' *                                9 : cad drawing                          *
' *            sDir             = directory location of database        *
' *            sName            = name of database                      *
' *            aFCtype          = feature class type (only used for     *
' *                                coverages, access databases and CAD) *
' *                                if not to be used specify as NULL,   *
' *                                for opmode = 5 this is the name of   *
' *                                the feature class to be opened       *
' *                                for opmode = 6 this is the name of   *
' *                                a dataset to be opened and sName is  *
' *                                the name of the access database      *
' *                                for opmode = 9 this is the name of   *
' *                                the feature class to be opened, valid*
' *                                values for this mode include POINT,  *
' *                                POLYLINE, POLYGON and ANNOTATION     *
' *                                                                     *
' *  RETURN:   avOpenFeatClass = dataset that is opened, if none, the   *
' *                                value will be set to NOTHING         *
' *                                                                     *
' *  Dim opmode As Integer                                              *
' *  Dim sDir As String, sName As String, aFCtype As String             *
' *  Dim avOpenFeatClass As IUnknown                                    *
' *                                                                     *
Public Function avOpenWorkspace(opmode, sDir, sName) As IWorkspace
' *                                                                     *
' *  PURPOSE:  OPEN A WORKSPACE FOR PROCESSING                          *
' *                                                                     *
' *  GIVEN:    opmode           = type of workspace to be opened        *
' *                                1 : shapefile                        *
' *                                2 : raster                           *
' *                                3 : tin                              *
' *                                4 : coverage                         *
' *                                5 : access database                  *
' *            sDir             = directory location of workspace       *
' *            sName            = name of workspace                     *
' *                                                                     *
' *  RETURN:   avOpenWorkspace = workspace that is opened, if none,     *
' *                                the value will be set to NOTHING     *
' *                                                                     *
' *  Dim opmode As Integer                                              *
' *  Dim sDir As String, sName As String                                *
' *  Dim avOpenWorkspace As IWorkspace                                  *
' *                                                                     *
Public Sub avPaletteGetList(aGalleryType, _
                            thePalette As IStyleGallery, aGalleryList)
' *                                                                     *
' *  PURPOSE:  GET A LIST OF THE ITEMS WITHIN A SPECIFIC GALLERY        *
' *                                                                     *
' *  GIVEN:    aGalleryType = type of gallery to be processed           *
' *                             keyword       name of gallery           *
' *                             PEN         : Line Symbols              *
' *                             MARKER      : Marker Symbols            *
' *                             FILL        : Fill Symbols              *
' *                             TEXT        : Text Symbols              *
' *                             COLOR       : Colors                    *
' *                             REFERENCE   : Reference Systems         *
' *                             SHADOWS     : Shadows                   *
' *                             AREAPATCHES : Area Patches              *
' *                             LINEPATCHES : Line Patches              *
' *                             LABELS      : Labels                    *
' *                             NORTHARROWS : North Arrows              *
' *                             SCALEBARS   : Scale Bars                *
' *                             LEGENDITEMS : Legend Items              *
' *                             SCALETEXT   : Scale Texts               *
' *                             COLORRAMPS  : Color Ramps               *
' *                             BORDERS     : Borders                   *
' *                             BACKGROUNDS : Backgrounds               *
' *            thePalette   = style gallery, contains all galleries     *
' *                                                                     *
' *  RETURN:   aGalleryList = list of items in the desired gallery      *
' *                                                                        *
' *   Dim aGalleryType As String, thePalette As IStyleGallery              *
' *  Dim aGalleryList As New Collection                                    *
' *                                                                        *
Public  Sub  avPaletteGetNames(aGalleryType,  _
                               thePalette As IStyleGallery, aNameList)
' *                                                                        *
' *   PURPOSE:   GET A LIST OF THE NAMES OF THE ITEMS WITHIN A SPECIFIC   *
' *              GALLERY                                                   *
' *                                                                        *
' *   GIVEN:     aGalleryType = type of gallery to be processed            *
' *                              keyword       name of gallery             *
' *                             PEN          : Line Symbols                *
' *                             MARKER       : Marker Symbols              *
' *                             FILL         : Fill Symbols                *
' *                             TEXT         : Text Symbols                *
' *                             COLOR        : Colors                      *
' *                              REFERENCE   : Reference Systems           *
' *                             SHADOWS      : Shadows                     *
' *                              AREAPATCHES : Area Patches                *
' *                              LINEPATCHES : Line Patches                *
' *                             LABELS       : Labels                      *
' *                              NORTHARROWS : North Arrows                *
' *                             SCALEBARS    : Scale Bars                  *
' *                              LEGENDITEMS : Legend Items                *
' *                             SCALETEXT    : Scale Texts                 *
' *                             COLORRAMPS   : Color Ramps                 *
' *                             BORDERS      : Borders                     *
' *                              BACKGROUNDS : Backgrounds                 *
' *               thePalette    = style gallery, contains all galleries    *
' *                                                                        *
' *   RETURN:    aNameList     = list of names corresponding to the        *
' *                              items in the desired gallery              *
' *                                                                        *
' *   NOTE:       There is a one to one correspondance between the names  *
' *               in aNameList and the symbols in aGalleryList passed      *
' *               by avPaletteGetList. Use this subroutine when it is      *
' *              desired to know the name of a symbol in a gallery         *
' *                                                                        *
' *   Dim aGalleryType As String, thePalette As IStyleGallery              *
' *  Dim aNameList As New Collection                                       *
' *                                                                        *
Public  Sub  avPanTo(pmxDoc  As  IMxDocument,  thePoint  As  IPoint)
' *                                                                        *
' *   PURPOSE:   SCRIPT TO CENTER THE DISPLAY ABOUT A POINT                *
' *                                                                        *
' *  GIVEN:     pmxDoc    = the active view                                *
' *               thePoint = point that the display is to be centered      *
' *                          about                                         *
' *                                                                        *
' *  RETURN:    nothing                                                    *
' *                                                                        *
' *  Dim pmxDoc As IMxDocument                                             *
' *  Dim thePoint As IPoint                                                *
' *                                                                        *
Public  Function  avPix2Map(pixelUnits  As  Double)  As  Double
' *                                                                        *
' *   PURPOSE:  TO CONVERT PIXELS INTO MAP UNITS                           *
' *                                                                        *
' *  GIVEN:     pixelUnits = number of pixels                              *
' *                                                                        *
' *   RETURN:    avPix2Map   = map units value representing the number    *
' *                             of pixels                                  *
' *                                                                        *
' *   Dim pixelUnits As Double, avPix2Map As Double                        *
' *                                                                        *
Public  Sub  avPlAsList(pFeatureGeom  As  IGeometry,  shapeList)
' *                                                                        *
' *   PURPOSE:  TO CREATE A POINT LIST FROM A GEOMETRY OBJECT              *
' *                                                                       *
' *   GIVEN:     pFeatureGeom = geometry object to be processed           *
' *                                                                       *
' *   RETURN:    shapeList     = list of points comprising the object     *
' *                              structure of shapeList is:               *
' *                               Item 1: collection of points in part 1  *
' *                              Repeat Item 1 for each part              *
' *                              So that, shapeList is a list of          *
' *                              collections with each collection         *
' *                              containing points                        *
' *                                                                       *
' *   NOTE:      (a) This routine can be used for Point, Polyline,        *
' *                  Polygon and Multi-point features                     *
' *                (b) Use subroutine avAsList when an IFeature object is *
' *                  known and not an IGeometry object                    *
' *                                                                       *
' *   Dim pFeatureGeom As IGeometry                                       *
' *   Dim shapeList As New Collection                                     *
' *                                                                       *
Public  Sub  avPlAsList2(pFeatureGeom  As  IGeometry,  shapeList)
' *                                                                       *
' *   PURPOSE:   TO CREATE A POINT LIST FROM A GEOMETRY OBJECT            *
' *                                                                       *
' *   GIVEN:     pFeatureGeom = geometry object to be processed           *
' *                                                                       *
' *   RETURN:    shapeList     = list of points comprising the object     *
' *                              structure of shapeList is:               *
' *                              Item 1: number of parts                  *
' *                               Item 2: number of points in part 1      *
' *                               Item 3: x value of point 1 in part 1    *
' *                               Item 4: y value of point 1 in part 1    *
' *                               Item 5: z value of point 1 in part 1    *
' *                               Item 6: m value of point 1 in part 1    *
' *                               Item 7: id value of point 1 in part 1   *
' *                               Item 8: Repeat Items 3 - 7 for each     *
' *                                       point                           *
' *                              Repeat Items 2 - 8 for each part         *
' *                                                                       *
' *   NOTE:       This routine can be used for Point, Polyline, Polygon   *
' *              and Multi-point features                                 *
' *                                                                       *
' *   Dim pFeatureGeom As IGeometry                                       *
' *   Dim shapeList As New Collection                                     *
' *                                                                       *
Public  Sub  avPlFindMinSeg(elmntList,  thePart,  minDis)
' *                                                                       *
' *   PURPOSE:   TO FIND THE SMALLEST SEGMENT LENGTH IN A POLYLINE OR     *
' *              POLYGON SHAPE                                            *
' *                                                                       *
' *   GIVEN:     elmntList = list of points comprising the feature        *
' *                          structure of elmntList is:                   *
' *                          Item 1: number of parts                      *
' *                           Item 2: number of points in part 1          *
' *                           Item 3: x value of point 1 in part 1        *
' *                           Item 4: y value of point 1 in part 1        *
' *                           Item 5: z value of point 1 in part 1        *
' *                           Item 6: m value of point 1 in part 1        *
' *                           Item 7: id value of point 1 in part 1       *
' *                           Item 8: Repeat Items 3 - 7 for each         *
' *                                   point                               *
' *                          Repeat Items 2 - 8 for each part             *
' *                                                                       *
' *   RETURN:    thePart    = the part of the polyline (0 - i)            *
' *                           part numbers begin at zero, not one         *
' *              minDis     = the smallest segment length in the shape    *
' *                                                                       *
' *   Dim elmntList As New Collection                                     *
' *   Dim thePart As Long, minDis As Double                               *
' *                                                                       *
```

```
Public Sub avPlFindVertex(ipmode, elmntList, X, Y, thePart, thePt)
' *                                                                  *
' *   PURPOSE:  TO FIND THE VERTEX MATCHING OR CLOSEST TO A LOCATION *
' *                                                                  *
' *  GIVEN:    ipmode    = the mode of operation                     *
' *                          0 : find first vertex matching a location *
' *                          1 : find the vertex closest to a location *
' *             elmntList = list of points comprising the feature    *
' *                         structure of elmntList is:               *
' *                         Item 1: number of parts                  *
' *                          Item 2: number of points in part 1      *
' *                          Item 3: x value of point 1 in part 1    *
' *                          Item 4: y value of point 1 in part 1    *
' *                          Item 5: z value of point 1 in part 1    *
' *                          Item 6: m value of point 1 in part 1    *
' *                          Item 7: id value of point 1 in part 1   *
' *                          Item 8: Repeat Items 3 - 7 for each     *
' *                                  point                           *
' *                         Repeat Items 2 - 8 for each part         *
' *            X,Y        = coordinates of new point                 *
' *                                                                  *
' *  RETURN:   thePart    = the part of the polyline (0 - i)         *
' *                          part numbers begin at zero, not one     *
' *             thePt      = point number in part closest to location *
' *                          point numbers begin at one, not zero    *
' *                                                                  *
' *  Dim ipmode As Integer                                           *
' *   Dim elmntList As New Collection, X As Double, Y As Double      *
' *  Dim thePart, thePt As Long                                      *
' *                                                                  *
Public Sub avPlGet3Pt(shapeList, thePart, X1, Y1, Xm, Ym, X2, Y2)
' *                                                                  *
' *   PURPOSE:  TO GET 3 POINTS FROM A FEATURE POINT LIST FOR A      *
' *             SPECIFIC PART IN THE FEATURE                         *
' *                                                                  *
' *   GIVEN:    shapeList = list of points comprising the feature    *
' *                         structure of shapeList is:               *
' *                         Item 1: number of parts                  *
' *                          Item 2: number of points in part 1      *
' *                          Item 3: x value of point 1 in part 1    *
' *                          Item 4: y value of point 1 in part 1    *
' *                          Item 5: z value of point 1 in part 1    *
' *                          Item 6: m value of point 1 in part 1    *
' *                          Item 7: id value of point 1 in part 1   *
' *                          Item 8: Repeat Items 3 - 7 for each     *
' *                                  point                           *
' *                         Repeat Items 2 - 8 for each part         *
' *             thePart    = the part of the polyline (0 - i)        *
' *                          part numbers begin at zero, not one     *
' *                                                                  *
' *  RETURN:   X1,Y1      = start point coordinates of part          *
' *             XM,YM      = mid point coordinates of part           *
' *             X2,Y2      = end point coordinates of part           *
' *                                                                  *
' *  Dim shapeList As New Collection, thePart As Long                *
' *  Dim X1, Y1, XM, YM, X2, Y2 As Double                            *
' *                                                                  *
Public Sub avPlModify(ipmode, elmntList, thePart, iPt, x, y, Z, newList)
' *                                                                  *
' *   PURPOSE:  TO MODIFY A SPECIFIC PART IN A FEATURE POINT LIST    *
' *                                                                  *
' *  GIVEN:    ipmode    = the mode of operation                     *
' *                          0 = change coordinates of a point       *
' *                         1 = insert new point                     *
' *                         2 = delete point                         *
' *             elmntList = list of points comprising the feature    *
' *                         structure of elmntList is:               *
' *                         Item 1: number of parts                  *
' *                          Item 2: number of points in part 1      *
' *                             Item 3: x value of point 1 in part 1        *
' *                             Item 4: y value of point 1 in part 1        *
' *                             Item 5: z value of point 1 in part 1        *
' *                             Item 6: m value of point 1 in part 1        *
' *                             Item 7: id value of point 1 in part 1       *
' *                             Item 8: Repeat Items 3 - 7 for each         *
' *                                     point                               *
' *                             Repeat Items 2 - 8 for each part            *
' *             thePart    = the part of the polyline (0 - i)               *
' *                             part numbers begin at zero, not one         *
' *             iPt        = index in part to be processed                  *
' *                             index numbers begin at one, not zero        *
' *                             if zero is specified the last point in the *
' *                             part will be processed                      *
' *             X,Y,Z      = coordinates of new point                       *
' *                                                                         *
' *   RETURN:    newList    = new list of points comprising the feature     *
' *                                                                         *
' *  Dim ipmode As Integer                                                  *
' *  Dim elmntList As New Collection, thePart As Long, iPt As Long          *
' *  Dim X As Double, Y As Double, Z As Double                              *
' *  Dim newList As New Collection                                          *
' *                                                                         *
Public Function avPointIDMake(xPt, yPt, IDPt) As IPoint
' *                                                                         *
' *   PURPOSE:  TO CREATE A POINT WITH ID OBJECT FROM COORDINATES           *
' *                                                                         *
' *  GIVEN:     xPt             = x coordinate of point                     *
' *             yPt             = y coordinate of point                     *
' *             IDPt            = ID value of point                         *
' *                                                                         *
' *  RETURN:    avPointIDMake = the point feature                           *
' *                                                                         *
' *  Dim xPt As Double, yPt As Double, IDPt As Long                         *
' *  Dim avPointIDMake As IPoint                                            *
' *                                                                         *
Public Function avPointMake(xPt, yPt) As IPoint
' *                                                                         *
' *   PURPOSE:  TO CREATE A POINT OBJECT FROM COORDINATES                   *
' *                                                                         *
' *  GIVEN:     xPt           = x coordinate of point                       *
' *             yPt           = y coordinate of point                       *
' *                                                                         *
' *  RETURN:    avPointMake = the point feature                             *
' *                                                                         *
' *  Dim xPt, yPt As Double                                                 *
' *  Dim avPointMake As IPoint                                              *
' *                                                                         *
Public Function avPointMMake(xPt, yPt, mPt) As IPoint
' *                                                                         *
' *   PURPOSE:  TO CREATE A POINT WITH M OBJECT FROM COORDINATES            *
' *                                                                         *
' *  GIVEN:     xPt            = x coordinate of point                      *
' *             yPt            = y coordinate of point                      *
' *             mPt            = m value of point                           *
' *                                                                         *
' *  RETURN:    avPointMMake = the point feature                            *
' *                                                                         *
' *  Dim xPt As Double, yPt As Double, mPt As Double                        *
' *  Dim avPointMMake As IPoint                                             *
' *                                                                         *
Public Sub avPointSetID(aPt As IPoint, IDPt)
' *                                                                         *
' *  PURPOSE:  TO SET THE ID ATTRIBUTE FOR A POINT                          *
' *                                                                         *
' *  GIVEN:     aPt      = point object to be modified                      *
' *             IDPt     = ID value of point                                *
' *                                                                         *
' *  RETURN:   nothing                                                      *
' *                                                                    *
' *  NOTE:      This subroutine modifies the aPt point object          *
' *                                                                    *
' *  Dim aPt As IPoint                                                 *
' *  Dim IDPt As Long                                                  *
' *                                                                    *
```

Public Sub avPointSetM(aPt As IPoint, mPt)

```
' *                                                                    *
' *  PURPOSE:  TO SET THE M ATTRIBUTE FOR A POINT                      *
' *                                                                    *
' *  GIVEN:    aPt      = point object to be modified                  *
' *            mPt      = m value of point                             *
' *                                                                    *
' *  RETURN:   nothing                                                 *
' *                                                                    *
' *  NOTE:      This subroutine modifies the aPt point object          *
' *                                                                    *
' *  Dim aPt As IPoint                                                 *
' *  Dim mPt As Double                                                 *
' *                                                                    *
```

Public Sub avPointSetZ(aPt As IPoint, zPt)

```
' *                                                                    *
' *  PURPOSE:  TO SET THE Z ATTRIBUTE FOR A POINT                      *
' *                                                                    *
' *  GIVEN:    aPt      = point object to be modified                  *
' *            zPt      = z coordinate of point                        *
' *                                                                    *
' *  RETURN:   nothing                                                 *
' *                                                                    *
' *  NOTE:      This subroutine modifies the aPt point object          *
' *                                                                    *
' *  Dim aPt As IPoint                                                 *
' *  Dim zPt As Double                                                 *
' *                                                                    *
```

Public Function avPointZMake(xPt, yPt, zPt) As IPoint

```
' *                                                                    *
' *  PURPOSE:  TO CREATE A 3D POINT OBJECT FROM COORDINATES            *
' *                                                                    *
' *  GIVEN:    xPt           = x coordinate of point                   *
' *            yPt           = y coordinate of point                   *
' *            zPt           = z coordinate of point                   *
' *                                                                    *
' *  RETURN:   avPointMakeZ = the point feature                        *
' *                                                                    *
' *  Dim xPt, yPt, zPt As Double                                       *
' *  Dim avPointMake As IPoint                                         *
' *                                                                    *
```

Public Function avPolygonMake(shapeList) As IPolygon

```
' *                                                                    *
' *  PURPOSE:  TO CREATE A POLYGON OBJECT FROM A POINT LIST            *
' *                                                                    *
' *  GIVEN:    shapeList      = the list of points comprising the      *
' *                             polygon                                *
' *                             structure of shapeList is:             *
' *                              Item 1: collection of points in part 1 *
' *                             Repeat Item 1 for each part            *
' *                             So that, shapeList is a list of        *
' *                             collections with each collection       *
' *                             containing points                      *
' *                                                                    *
' *  RETURN:   avPolygonMake = the polygon feature                     *
' *                                                                    *
' *  NOTE:     If the last point in a part is not the same as the      *
' *             first point in the part, a point will be added to make *
' *            sure that the part forms a closed polygon               *
' *                                                                    *
' *  Dim shapeList As New Collection                                   *
' *  Dim avPolygonMake As IPolygon                                     *
' *                                                                    *
```

```
Public  Function  avPolygonMake2(shapeList)  As  IPolygon
' *                                                                      *
' *   PURPOSE:   TO CREATE A POLYGON OBJECT FROM A POINT LIST            *
' *                                                                      *
' *   GIVEN:     shapeList       = the list of points comprising the     *
' *                                 polygon                              *
' *                                 structure of shapeList is:           *
' *                                 Item 1: number of parts              *
' *                                  Item 2: number of points in part 1  *
' *                                  Item 3: x value of point 1 in part 1 *
' *                                  Item 4: y value of point 1 in part 1 *
' *                                  Item 5: z value of point 1 in part 1 *
' *                                  Item 6: m value of point 1 in part 1 *
' *                                  Item 7: id value of point 1 in part 1 *
' *                                  Item 8: Repeat Items 3 - 7 for each *
' *                                          point                       *
' *                                 Repeat Items 2 - 8 for each part     *
' *                                                                      *
' *   RETURN:    avPolygonMake2 = the polygon feature                    *
' *                                                                      *
' *   NOTE:      If the last point in a part is not the same as the      *
' *                first point in the part, a point will be added to make *
' *              sure that the part forms a closed polygon               *
' *                                                                      *
' *   Dim shapeList As New Collection                                    *
' *   Dim avPolygonMake2 As IPolygon                                     *
' *                                                                      *
Public  Function  avPolyline2Pt(X1,  Y1,  X2,  Y2)  As  IPolyline
' *                                                                      *
' *   PURPOSE:   TO CREATE A TWO-POINT POLYLINE FROM COORDINATES         *
' *                                                                      *
' *   GIVEN:     X1              = x coordinate of start point           *
' *              Y1              = y coordinate of start point           *
' *              X2              = x coordinate of end point             *
' *              Y2              = y coordinate of end point             *
' *                                                                      *
' *   RETURN:    avPolyline2Pt = the polyine feature                     *
' *                                                                      *
' *   Dim X1, Y1, X2, Y2 As Double                                       *
' *   Dim avPolyline2Pt As IPolyline                                     *
' *                                                                      *
Public  Function  avPolylineMake(shapeList)  As  IPolyline
' *                                                                      *
' *   PURPOSE:   TO CREATE A POLYLINE OBJECT FROM A POINT LIST           *
' *                                                                      *
' *   GIVEN:     shapeList       = the list of points comprising the     *
' *                                 polyline                             *
' *                                 structure of shapeList is:           *
' *                                  Item 1: collection of points in part 1*
' *                                 Repeat Item 1 for each part          *
' *                                 So that, shapeList is a list of      *
' *                                  collections with each collection    *
' *                                 containing points                    *
' *                                                                      *
' *   RETURN:    avPolylineMake = the polyline feature                   *
' *                                                                      *
' *   Dim shapeList As New Collection                                    *
' *   Dim avPolylineMake As IPolyline                                    *
' *                                                                      *
Public  Function  avPolylineMake2(shapeList)  As  IPolyline
' *                                                                      *
' *   PURPOSE:   TO CREATE A POLYLINE OBJECT FROM A POINT LIST           *
' *                                                                      *
' *   GIVEN:     shapeList        = the list of points comprising the    *
' *                                  polyline                            *
' *                                   structure of shapeList is:         *
' *                                  Item 1: number of parts             *
' *                                   Item 2: number of points in part 1 *
' *                                   Item 3: x value of point 1 in part 1 *
' *                                Item 4: y value of point 1 in part 1 *
' *                                Item 5: z value of point 1 in part 1 *
' *                                Item 6: m value of point 1 in part 1 *
' *                                Item 7: id value of point 1 in part 1*
' *                                Item 8: Repeat Items 3 - 7 for each  *
' *                                      point                          *
' *                               Repeat Items 2 - 8 for each part      *
' *                                                                     *
' *  RETURN:    avPolylineMake2 = the polyline feature                  *
' *                                                                     *
' *  Dim shapeList As New Collection                                    *
' *  Dim avPolylineMake2 As IPolyline                                   *
' *                                                                     *
Public  Sub  avQuantile(pmxDoc  As  IMxDocument,  theTheme,  aField,  numClass)
' *                                                                     *
' *  PURPOSE:  TO SET THE LEGEND THAT IS ASSOCIATED WITH A THEME        *
' *              TO BE OF QUANTILE TYPE WITH THE CLASSES DETERMINED BY  *
' *             USING A QUANTILE METHOD                                 *
' *                                                                     *
' *  GIVEN:     pmxDoc     = the active view                            *
' *              theTheme  = theme to be processed                      *
' *               aField     = field name that theme is to be classified *
' *                          upon                                       *
' *              numClass  = number of classes to be generated          *
' *                                                                     *
' *  RETURN:    nothing                                                 *
' *                                                                     *
' *  NOTE:      (a) Divides the features in the theme into numClass     *
' *                  classifications of equal size using the values in  *
' *                  aField. This is only supported for numeric fields  *
' *              (b) After a theme has been classified you must use     *
' *                  avGetLegend to get the new legend that reflects    *
' *                  the new classification if you wish to manipulate   *
' *                  the labels or symbols in the classification        *
' *                                                                     *
' *  Dim pmxDoc As IMxDocument                                          *
' *  Dim theTheme As Variant                                            *
' *  Dim aField As String                                               *
' *  Dim numClass As Long                                               *
' *                                                                     *
Public  Sub  avQuantileX(pmxDoc  As  IMxDocument,  _
                              theTheme, aField, numClass, method)
' *                                                                     *
' *  PURPOSE:  TO SET THE LEGEND THAT IS ASSOCIATED WITH A THEME        *
' *            TO BE OF QUANTILE TYPE                                   *
' *                                                                     *
' *  GIVEN:     pmxDoc     = the active view                            *
' *              theTheme  = theme to be processed                      *
' *               aField     = field name that theme is to be classified *
' *                          upon                                       *
' *              numClass  = number of classes to be generated          *
' *             method    = classification method                       *
' *                            1 : Defined Interval (not implemented)   *
' *                          2 : Equal Interval                         *
' *                          3 : Natural Breaks                         *
' *                          4 : Quantile                               *
' *                            5 : Standard Deviation (not implemented) *
' *                                                                     *
' *  RETURN:    nothing                                                 *
' *                                                                     *
' *  NOTE:      (a) Divides the features in the theme into numClass     *
' *                  classifications using the values in aField. This   *
' *                  is only supported for numeric fields               *
' *              (b) After a theme has been classified you must use     *
' *                  avGetLegend to get the new legend that reflects    *
' *                  the new classification if you wish to manipulate   *
' *                  the labels or symbols in the classification        *
' *                                                                     *
' *  Dim pmxDoc As IMxDocument                                          *
' *  Dim theTheme As Variant                                                *
' *  Dim aField As String                                                   *
' *  Dim numClass As Long                                                   *
' *  Dim method As Long                                                     *
' *                                                                         *
Public Sub avQuery(pmxDoc As IMxDocument, theTheme, _
                       aQueryString, selSet As ISelectionSet, setType)
' *                                                                         *
' *  PURPOSE:  TO APPLY A QUERY TO A THEME OR A TABLE                       *
' *                                                                         *
' *  GIVEN:    pmxDoc       = the active view                               *
' *              theTheme     = name of theme or table to be processed     *
' *             aQueryString = query string to be applied                  *
' *                             Sample String field query for a            *
' *                            Shapefile:                                   *
' *                              aQueryStr = """PTCODE""" + " = 'BBBB'"    *
' *                             for a Personal geodatabase:                 *
' *                             aQueryStr = "PTCODE = 'bbbb'"               *
' *                             Sample Numeric field query for a            *
' *                              Shapefile and a Personal geodatabase       *
' *                            aQueryStr = "SLN >= 10"                      *
' *             selSet       = theme selection set                          *
' *             setType      = type of selection desired                    *
' *                             "NEW" : new selection set                   *
' *                             "ADD" : add to selection set                *
' *                              "AND" : select from selection set          *
' *                                                                         *
' *  RETURN:   nothing                                                      *
' *                                                                         *
' *  NOTE:      (a) Use avGetSelection to get the selection set that       *
' *                 contains the result of this query                      *
' *              (b) The query is applied even if the theme or layer       *
' *                  is set to be not selectable in ArcMap                 *
' *               (c) String queries on shapefiles are case sensitive,     *
' *                   while for personal geodatabases they are case        *
' *                 insensitive                                            *
' *                                                                         *
' *  Dim pmxDoc As IMxDocument                                              *
' *  Dim theTheme As Variant                                                *
' *  Dim aQueryString As String                                             *
' *  Dim selSet As ISelectionSet                                            *
' *  Dim setType As String                                                  *
' *                                                                         *
Public Function avRectMake4Pt(X1, Y1, X2, Y2, _
                                    X3, Y3, X4, Y4) As IPolygon
' *                                                                         *
' *   PURPOSE:  TO CREATE AN INCLINED POLYGON FROM COORDINATES OF FOUR     *
' *              POINTS THAT REPRESENT THE CORNERS OF THE POLYGON          *
' *                                                                         *
' *  GIVEN:    X1              = x coordinate of corner point 1            *
' *             Y1              = y coordinate of corner point 1            *
' *             X2              = x coordinate of corner point 2            *
' *             Y2              = y coordinate of corner point 2            *
' *             X3              = x coordinate of corner point 3            *
' *             Y3              = y coordinate of corner point 3            *
' *             X4              = x coordinate of corner point 4            *
' *             Y4              = y coordinate of corner point 4            *
' *                                                                         *
' *  RETURN:   avRectMake4PT = the polygon feature                          *
' *                                                                         *
' *   NOTE:      The corner points are connected in series in the order    *
' *               in which they are specified. A clockwise or counter-     *
' *             clockwise direction can be used                            *
' *                                                                         *
' *  Dim X1, Y1, X2, Y2, X3, Y3, X4, Y4 As Double                           *
' *  Dim avRectMake4Pt As IPolygon                                          *
' *                                                                         *
```

```
Public  Function  avRectMakeXY(X1,  Y1,  X2,  Y2)  As  IPolygon
' *                                                                    *
' *   PURPOSE:   TO CREATE A RECTANGULAR POLYGON FROM COORDINATES OF A  *
' *              DIAGONAL OF THE POLYGON                               *
' *                                                                    *
' *  GIVEN:     X1             = x coordinate of diagonal start point  *
' *              Y1             = y coordinate of diagonal start point  *
' *              X2             = x coordinate of diagonal end point    *
' *              Y2             = y coordinate of diagonal end point    *
' *                                                                    *
' *  RETURN:    avRectMakeXY = the polygon feature                     *
' *                                                                    *
' *  Dim X1, Y1, X2, Y2 As Double                                      *
' *  Dim avRectMakeXY As IPolygon                                      *
' *                                                                    *
Public  Sub  avRemoveDoc(aDocName)
' *                                                                    *
' *   PURPOSE:   REMOVE THE SPECIFIED LAYER OR TABLE FROM THE TABLE OF  *
' *               CONTENTS (DOES NOT DELETE ANY FILES FROM DISK)       *
' *                                                                    *
' *   GIVEN:     aDocName = name of theme or table to be removed from  *
' *                          the Table of Contents                     *
' *                                                                    *
' *  RETURN:    nothing                                                *
' *                                                                    *
' *   NOTE:      If the theme or table can not be found, the Table of  *
' *               Contents is left unaltered and no error is generated *
' *                                                                    *
' *  Dim aDocName As String                                            *
' *                                                                    *
Public  Sub  avRemoveDupStrings(theColl,  caseFlag)
' *                                                                    *
' *   PURPOSE:   REMOVE DUPLICATE STRINGS OR NUMBERS FROM A COLLECTION  *
' *                                                                    *
' *   GIVEN:     theColl   = collection containing strings from which  *
' *                          any duplicates will be removed            *
' *               caseFlag = flag denoting whether the collection is to *
' *                           be processed as case sensitive or case   *
' *                            insensitive (upper/lower case characters *
' *                          are treated the same)                     *
' *                            true = case sensitive, false = insensitive *
' *                                                                    *
' *  RETURN:    nothing                                                *
' *                                                                    *
' *  NOTE:      (a) theColl is changed by this subroutine              *
' *               (b) this subroutine will work for collections that   *
' *                   contain numbers, as well as, strings             *
' *               (c) if theColl contains numbers, not strings, set the *
' *                    caseFlag to be true, if it is false an error will *
' *                  be generated                                      *
' *               (d) if theColl contains numbers and strings, set the  *
' *                    caseFlag to be true, if it is false an error will *
' *                  be generated                                      *
' *                                                                    *
' *  Dim theColl As New Collection                                     *
' *  Dim caseFlag As Boolean                                           *
' *                                                                    *
Public  Function  avRemoveFields(pmxDoc  As  IMxDocument,  _
                                  theTheme, theFieldS)
' *                                                                    *
' *  PURPOSE:  TO REMOVE FIELDS FROM A LAYER OR TABLE                  *
' *                                                                    *
' *  GIVEN:     pmxDoc          = the active view                      *
' *              theTheme        = the theme or table to be processed  *
' *              theFields       = list of fields to be removed, the   *
' *                                 items in this list are index values *
' *                                 for the fields to be deleted, they *
' *                                 are numeric values not objects     *
' *                                                                    *
' *   RETURN:    avRemoveFields = error flag (0 = no error, 1 = error)   *
' *                                                                       *
' *   NOTE:      (a) In order to remove fields from a layer or table     *
' *                  the editor can not be in an edit state, this         *
' *                  routine will stop the editor, saving any changes    *
' *                  that may have been made, prior to removing the      *
' *                  fields                                              *
' *              (b) If an invalid index value appears in the list, -1, *
' *                  it will be ignored (an error is not generated)      *
' *              (c) Do not use this routine to delete the SHAPE field   *
' *                                                                       *
' *   Dim pmxDoc As IMxDocument                                          *
' *   Dim theTheme As Variant                                            *
' *   Dim theFields As New Collection                                    *
' *   Dim avRemoveFields As Integer                                      *
' *                                                                       *
```

```
Public Sub avRemoveGraphic(pElement As IElement)
' *                                                                       *
' *   PURPOSE:  TO DELETE A GRAPHIC ELEMENT FROM THE DISPLAY             *
' *                                                                       *
' *   GIVEN:    pElement = graphic to be deleted                          *
' *                                                                       *
' *   RETURN:   nothing                                                   *
' *                                                                       *
' *   Dim pElement As IElement                                           *
' *                                                                       *
```

```
Public Sub avRemoveRecord(pmxDoc As IMxDocument, theTheme, theRcrd)
' *                                                                       *
' *   PURPOSE:  DELETE A RECORD OR THE SELECTED FEATURES (ROWS) IN A     *
' *             LAYER OR TABLE                                            *
' *                                                                       *
' *   GIVEN:    pmxDoc   = the active view                                *
' *             theTheme = the theme or table to be processed            *
' *             theRcrd  = mode of deletion                               *
' *                        >= 0 : record of feature (row) for deletion *
' *                        = -1 : delete selected features (rows) in a *
' *                               theme or table, if there are no        *
' *                               selected features (rows), nothing      *
' *                               will be deleted                        *
' *                        = -2 : delete all features in a theme or      *
' *                               table regardless of the current        *
' *                               selection set                          *
' *                                                                       *
' *   RETURN:   nothing                                                   *
' *                                                                       *
' *   NOTE:     The theme or table must be editable prior to deleting    *
' *             any features (rows) from the theme or table              *
' *                                                                       *
' *   Dim pmxDoc As IMxDocument                                          *
' *   Dim theTheme As Variant                                            *
' *   Dim theRcrd As Long                                                *
' *                                                                       *
```

```
Public Sub avResize(aDoc As IUnknown, aWidth, aHeight)
' *                                                                       *
' *   PURPOSE:  TO RESIZE A WINDOW OBJECT                                *
' *                                                                       *
' *   GIVEN:    aDoc    = the window object                               *
' *             aWidth  = the new width of the window                     *
' *             aHeight = the new height of the window                    *
' *                                                                       *
' *   RETURN:   nothing                                                   *
' *                                                                       *
' *   Dim aDoc As IUnknown                                               *
' *   Dim aWidth, aHeight As Long                                        *
' *                                                                       *
```

```
Public Function avReturnArea(theGeom As IGeometry) As Double
' *                                                                       *
' *   PURPOSE:  GET THE AREA OF A GEOMETRY                               *
' *                                                                       *
' *  GIVEN:     theGeom       = the geometry to be processed            *
' *                                                                     *
' *  RETURN:    avReturnArea = the area of the geometry                 *
' *                                                                     *
' *  NOTE:       If invalid geometry is specified, avReturnArea will    *
' *             be set to zero                                          *
' *                                                                     *
' *  Dim theGeom As IGeometry                                           *
' *  Dim avReturnArea As Double                                         *
' *                                                                     *
Public  Function  avReturnCenter(theGeom  As  IGeometry)  As  IPoint
' *                                                                     *
' *  PURPOSE:  GET THE CENTROID OF A GEOMETRY                           *
' *                                                                     *
' *  GIVEN:     theGeom         = the geometry to be processed          *
' *                                                                     *
' *  RETURN:    avReturnCenter = the centroid of the geometry           *
' *                                                                     *
' *  NOTE:       If invalid geometry is specified, avReturnCenter will  *
' *             be set to NOTHING                                       *
' *                                                                     *
' *  Dim theGeom As IGeometry                                           *
' *  Dim avReturnCenter As IPoint                                       *
' *                                                                     *
Public  Function  avReturnDifference(aShape1  As  IGeometry,  _
                                        aShape2 As IGeometry) As IGeometry
' *                                                                     *
' *  PURPOSE:  TO REMOVE FROM THE BASE SHAPE THE OVERLAP WITH A         *
' *             SECOND SHAPE TO FORM A NEW SHAPE                        *
' *                                                                     *
' *  GIVEN:     aShape1             = base shape                        *
' *              aShape2             = second shape whose overlap with  *
' *                                     the base shape will be removed  *
' *                                    from the base shape              *
' *                                                                     *
' *  RETURN:     avReturnDifference = new shape reflecting the removal  *
' *                                     of the overlap from the base shape*
' *                                                                     *
' *  NOTE:       If there is no overlap between the two shapes the      *
' *              shape that is passed back will be the same as the base *
' *             shape                                                   *
' *                                                                     *
' *  Dim aShape1 As IGeometry, aShape2 As IGeometry                     *
' *  Dim avReturnDifference As IGeometry                                *
' *                                                                     *
Public  Function  avReturnIntersection(aShape1  As  IGeometry,  _
                                          aShape2 As IGeometry) As IGeometry
' *                                                                     *
' *  PURPOSE:  TO INTERSECT TWO SHAPES TO FORM A NEW SHAPE              *
' *                                                                     *
' *  GIVEN:     aShape1               = base shape                      *
' *              aShape2               = second shape to be intersected *
' *                                     with the base shape             *
' *                                                                     *
' *  RETURN:    avReturnIntersection = new shape reflecting the         *
' *                                      intersection of the two shapes *
' *                                                                     *
' *  NOTE:       (a) If the shapes do not intersect an empty shape will *
' *                  be passed back                                     *
' *              (b) When dealing with polygon shapes make sure the     *
' *                   polygon is defined in a clockwise direction, if   *
' *                  not, an intersection may not be computed           *
' *                                                                     *
' *  Dim aShape1 As IGeometry, aShape2 As IGeometry                     *
' *  Dim avReturnIntersection As IGeometry                              *
' *                                                                     *
Public  Function  avReturnLength(theGeom  As  IGeometry)  As  Double
' *                                                                     *
' *  PURPOSE:  GET THE PERIMETER OR LENGTH OF A GEOMETRY                *
' *                                                                     *
' *   GIVEN:     theGeom          = the geometry to be processed        *
' *                                                                     *
' *   RETURN:    avReturnLength = the perimeter or length of the        *
' *                               geometry                              *
' *                                                                     *
' *   NOTE:       For multi-part features avReturnLength will be the    *
' *               total length, which includes all parts                *
' *                                                                     *
' *   Dim theGeom As IGeometry                                          *
' *   Dim avReturnLength As Double                                      *
' *                                                                     *
Public  Function  avReturnMerged(aShape1  As  IGeometry,  _
                                 aShape2 As IGeometry) As IGeometry
' *                                                                     *
' *   PURPOSE:   TO MERGE TWO SHAPES TOGETHER TO FORM A NEW SHAPE       *
' *                                                                     *
' *   GIVEN:     aShape1         = base shape                           *
' *               aShape2         = second shape to be merged with the  *
' *                                base shape                           *
' *                                                                     *
' *   RETURN:    avReturnMerged = new shape reflecting the merging      *
' *                                                                     *
' *   Dim aShape1 As IGeometry, aShape2 As IGeometry                    *
' *   Dim avReturnMerged As IGeometry                                   *
' *                                                                     *
Public  Sub  avReturnShared(aShape1  As  IGeometry,  _
                            aShape2 As IGeometry, aTol, _
                            X1, Y1, X2, Y2, _
                            commonSid As IPolyline, _
                            aNewShape As IGeometry)
' *                                                                     *
' *   PURPOSE:   TO CHANGE THE SIDE OF A SHAPE TO BE IDENTICAL TO THE   *
' *               SIDE IN ANOTHER SHAPE BASED UPON TWO COMMON VERTICES  *
' *                                                                     *
' *   GIVEN:     aShape1    = base shape                                *
' *               aShape2    = second shape to be altered to match      *
' *                            the common side with the base shape      *
' *               aTol       = proximity tolerance                      *
' *                            0 : denotes use the smallest segment     *
' *                                length between the two shapes        *
' *                                                                     *
' *   RETURN:    X1, Y1     = first matching coordinates of the common  *
' *                           side on the base shape                    *
' *               X2, Y2     = second matching coordinates of the common *
' *                           side on the base shape                    *
' *               commonSid = polyline representing the shared side     *
' *               aNewShape = new shape reflecting the incorporation of *
' *                            the common side into the second shape    *
' *                                                                     *
' *   NOTE:       If two matching vertices can not be found, the values *
' *               of commonSid and aNewShape will both be set to NOTHING *
' *                                                                     *
' *   Dim aShape1 As IGeometry, aShape2 As IGeometry, aTol As Double    *
' *   Dim X1 As Double, Y1 As Double, X2 As Double, Y2 As Double        *
' *   Dim commonSid As IPolyline, aNewShape As IGeometry                *
' *                                                                     *
Public  Function  avReturnUnion(aShape1  As  IGeometry,  _
                                aShape2 As IGeometry) As IGeometry
' *                                                                     *
' *   PURPOSE:   TO UNION TWO SHAPES TOGETHER TO FORM A NEW SHAPE       *
' *                                                                     *
' *   GIVEN:     aShape1        = base shape                            *
' *               aShape2        = second shape to be unioned with the  *
' *                               base shape                            *
' *                                                                     *
' *   RETURN:    avReturnUnion = new shape reflecting the unioning      *
' *                                                                     *
' *   Dim aShape1 As IGeometry, aShape2 As IGeometry                    *
' *  Dim avReturnUnion As IGeometry                                    *
' *                                                                    *
Public  Function  avReturnValue(pmxdoc  As  IMxDocument,  _
                                  theTheme, aField, aRecord) As Variant
' *                                                                    *
' *   PURPOSE:   TO RETRIEVE A VALUE IN A SPECIFIC FIELD OF A SPECIFIC  *
' *              ROW (RECORD) FOR A LAYER OR TABLE                      *
' *                                                                    *
' *  GIVEN:     pmxDoc         = the active view                        *
' *               theTheme       = the theme or table to be processed   *
' *               aField          = index value denoting field the desired *
' *                                 attribute is to be retrieved from   *
' *               aRecord        = record of the theme or table to be   *
' *                                processed                            *
' *                                                                    *
' *   RETURN:    avReturnValue = attribute that was retrieved (not      *
' *                               geometry but data, see note below)    *
' *                                                                    *
' *   NOTE:       (a) Do not use this routine to retrieve geometry from  *
' *                    the SHAPE field, use this routine to extract      *
' *                     attribute data only (see avGetFeature for how to *
' *                    extract the geometry of a feature)                *
' *                (b) If an error is detected avReturnValue will be set *
' *                  to NULL                                             *
' *                                                                    *
' *  Dim pmxDoc As IMxDocument                                          *
' *  Dim theTheme As Variant                                            *
' *  Dim aField, aRecord As Long                                        *
' *  Dim avReturnValue As Variant                                       *
' *                                                                    *
Public  Function  avReturnVisExtent(pDT  _
                                   As IDisplayTransformation) As IEnvelope
' *                                                                    *
' *  PURPOSE:  GET THE CURRENT EXTENT OF THE VIEW                       *
' *                                                                    *
' *  GIVEN:     pDT                  = the screen display transformation *
' *                                                                    *
' *   RETURN:    avReturnVisExtent = the current extent of the view     *
' *                                                                    *
' *  Dim pDT As IDisplayTransformation                                  *
' *  Dim avReturnVisExtent As IEnvelope                                 *
' *                                                                    *
Public  Sub  avSelectByFTab(pmxDoc  As  IMxDocument,  _
                             elmntTheme, seltrTheme,selMod, selTol, setType)
' *                                                                    *
' *   PURPOSE:   TO SELECT FEATURES IN A THEME BASED UPON THE SELECTED  *
' *              FEATURES IN ANOTHER THEME                              *
' *                                                                    *
' *  GIVEN:     pmxDoc      = the active view                           *
' *              elmntTheme = name of theme to be processed             *
' *              seltrTheme = selector theme to be used                 *
' *              selMod      = selection mode of operation              *
' *                             "INTERSECTS"                            *
' *                             "ISWITHINDISTANCEOF"                    *
' *                             "HASCENTERWITHIN"                       *
' *                             "ISCOMPLETELYWITHIN"                    *
' *               selTol      = selection distance tolerance (use 0 if  *
' *                              the shape of the features in the selector *
' *                              theme are to be used as is, no buffering) *
' *              setType     = type of selection desired                *
' *                             "NEW" : new selection set               *
' *                             "ADD" : add to selection set            *
' *                              "AND" : select from selection set      *
' *                                                                    *
' *  RETURN:   nothing                                                  *
' *                                                                    *
' *   NOTE:       Use avUpdateSelection to update the display of the    *
' *              selected features                                      *
' *                                                                    *
' *  Dim pmxDoc As IMxDocument                                          *
' *  Dim elmntTheme As Variant                                          *
' *  Dim seltrTheme As Variant                                          *
' *  Dim selMod As String                                               *
' *  Dim selTol As Double                                               *
' *  Dim setType As String                                              *
' *                                                                     *
Public  Sub  avSelectByPoint(pmxDoc  As  IMxDocument,  elmntTheme,  _
                             thePoint As IPoint, selTol, setType)
' *                                                                     *
' *   PURPOSE:   TO SELECT FEATURES IN A THEME BASED UPON A POINT       *
' *                                                                     *
' *  GIVEN:     pmxDoc      = the active view                           *
' *               elmntTheme = name of theme to be processed            *
' *               thePoint    = point object to be used in the search   *
' *               selTol      = selection distance tolerance            *
' *               setType     = type of selection desired               *
' *                              "NEW" : new selection set              *
' *                              "ADD" : add to selection set           *
' *                              "AND" : select from selection set      *
' *                                                                     *
' *  RETURN:    nothing                                                 *
' *                                                                     *
' *   NOTE:      Use avUpdateSelection to update the display of the     *
' *              selected features                                      *
' *                                                                     *
' *  Dim pmxDoc As IMxDocument                                          *
' *  Dim elmntTheme As Variant                                          *
' *  Dim thePoint As IPoint                                             *
' *  Dim selTol As Double                                               *
' *  Dim setType As String                                              *
' *                                                                     *
Public  Sub  avSelectByPolygon(pmxDoc  As  IMxDocument,  _
                                 elmntTheme, theGeom As IGeometry, setType)
' *                                                                     *
' *   PURPOSE:   TO SELECT FEATURES IN A THEME BASED UPON A POLYGON     *
' *                                                                     *
' *  GIVEN:     pmxDoc      = the active view                           *
' *               elmntTheme = name of theme to be processed            *
' *              theGeom     = geometry to be used                      *
' *               setType     = type of selection desired               *
' *                              "NEW" : new selection set              *
' *                              "ADD" : add to selection set           *
' *                              "AND" : select from selection set      *
' *                                                                     *
' *  RETURN:    nothing                                                 *
' *                                                                     *
' *   NOTE:      Use avUpdateSelection to update the display of the     *
' *              selected features                                      *
' *                                                                     *
' *  Dim pmxDoc As IMxDocument                                          *
' *  Dim elmntTheme As Variant                                          *
' *  Dim theGeom As IGeometry                                           *
' *  Dim setType As String                                              *
' *                                                                     *
Public  Sub  avSetActive(pmxDoc  As  IMxDocument,  theTheme,  sStatus)
' *                                                                     *
' *  PURPOSE:   TO MAKE A THEME SELECTABLE OR NOT                       *
' *                                                                     *
' *  GIVEN:     pmxDoc    = the active view                             *
' *              theTheme = theme to be processed                       *
' *               sStatus   = selectable status (True   = selectable)   *
' *                                              (False = not selectable)  *
' *                                                                     *
' *  RETURN:    nothing                                                 *
' *                                                                     *
' *  Dim pmxDoc As IMxDocument                                          *
' *  Dim theTheme As Variant                                            *
' *  Dim sStatus As Boolean                                             *
' *                                                                   *
Public Sub avSetAlias(col, anAlias)
' *                                                                   *
' *  PURPOSE:  ASSIGN AN ALIAS TO A FIELD FOR A LAYER OR TABLE       *
' *                                                                   *
' *  GIVEN:    col     = index value representing the field that an   *
' *                      alias is to be assigned to                   *
' *            anAlias = the alias to be assigned to the field        *
' *                                                                   *
' *  RETURN:   nothing                                                *
' *                                                                   *
' *  NOTE:     The current layer/table is processed, the subroutines  *
' *            avGetFTab or avGetVTab can be used to establish the    *
' *            current layer or table                                 *
' *                                                                   *
' *  Dim col As Long                                                  *
' *  Dim anAlias As String                                            *
' *                                                                   *
Public Sub avSetAll(pmxDoc As IMxDocument, theTheme, _
                    psTableSel As ISelectionSet)
' *                                                                   *
' *  PURPOSE:  TO SELECT ALL OF THE FEATURES OR ROWS FOR A LAYER OR   *
' *            TABLE                                                  *
' *                                                                   *
' *  GIVEN:    pmxDoc     = the active view                           *
' *            theTheme   = the theme or table to be processed        *
' *                                                                   *
' *  RETURN:   psTableSel = the selection set for the theme or table  *
' *                         with all features or rows selected        *
' *                                                                   *
' *  NOTE:     Following the call to this subroutine make a call to   *
' *            avUpdateSelection to update the display so that the    *
' *            selected features can be seen                          *
' *                                                                   *
' *  Dim pmxDoc As IMxDocument                                        *
' *  Dim theTheme As Variant                                          *
' *  Dim psTableSel As ISelectionSet                                  *
' *                                                                   *
Public Sub avSetEditable(pmxDoc As IMxDocument, theTheme, eStatus)
' *                                                                   *
' *  PURPOSE:  START OR TERMINATE THE EDITING ON A LAYER OR TABLE     *
' *                                                                   *
' *  GIVEN:    pmxDoc   = the active view                             *
' *            theTheme = theme or table to be processed              *
' *            eStatus  = editing status (True  = start editing)      *
' *                                      (False = stop editing)       *
' *                                                                   *
' *  RETURN:   nothing                                                *
' *                                                                   *
' *  NOTE:     (a) For layers the editing is not terminated but       *
' *                rather any buffered writes are simply flushed.     *
' *                For tables the editing is terminated.  To          *
' *                terminate the editing on layers use the subroutine *
' *                avStopEditing.                                     *
' *            (b) If the layer or table is to be made editable and   *
' *                the layer or table is already editable, no action  *
' *                will be taken and the layer or table will remain   *
' *                editable                                           *
' *                                                                   *
' *  Dim pmxDoc As IMxDocument                                        *
' *  Dim theTheme As Variant                                          *
' *  Dim eStatus As Boolean                                           *
' *                                                                   *
Public Sub avSetEditable2(aLayer As IFeatureLayer, eStatus)
' *                                                                   *
' *  PURPOSE:  START OR TERMINATE THE EDITING ON A LAYER              *
' *                                                                   *
' *  GIVEN:    aLayer  = layer to be processed                        *
' *            eStatus = editing status (True  = start editing)       *
' *                                       (False = stop editing)        *
' *                                                                     *
' *  RETURN:   nothing                                                  *
' *                                                                     *
' *  NOTE:      (a) Any edits that may have been made to the layer are *
' *                 committed to disk when eStatus is False             *
' *             (b) If the layer is to be made editable and the layer  *
' *                 is already editable, no action will be taken and   *
' *                 the layer will remain editable                      *
' *                                                                     *
' *  Dim aLayer As IFeatureLayer                                        *
' *  Dim eStatus As Boolean                                             *
' *                                                                     *
Public Sub avSetEditableTheme(pmxDoc As IMxDocument, theTheme, theType)
' *                                                                     *
' *  PURPOSE:  SCRIPT TO SET THE TYPE OF TASK FOR EDITING A THEME       *
' *                                                                     *
' *  GIVEN:    pmxDoc   = the active view                               *
' *            theTheme = the theme to be processed                     *
' *                       if NULL, editor will be stopped saving any   *
' *                       edits that may have been made                 *
' *            theType  = the type of task to be performed, if not     *
' *                       equal to zero will set the current task of   *
' *                       the editor                                    *
' *                       0    = stop sketch session                    *
' *                       1    = modify feature (this will start a     *
' *                              sketch session), for polylines and    *
' *                              polygons, handles will be drawn at    *
' *                              the vertices composing the feature    *
' *                       2    = create new feature                     *
' *                       9    = same as 0 except assign the current   *
' *                              sketch geometry to the feature that   *
' *                              is stored globally (ugLastFeatureSV) *
' *                       NULL = do nothing                             *
' *                                                                     *
' *  RETURN:   nothing                                                  *
' *                                                                     *
' *  NOTE:      The global variable ugSketch is used to keep track of  *
' *             whether a sketch session is active or not. If the      *
' *             value of ugSketch = 0, a sketch session is not active, *
' *             if ugSketch = 1, a sketch session is active             *
' *                                                                     *
' *  Dim pmxDoc As IMxDocument                                          *
' *  Dim theTheme As Variant                                            *
' *  Dim theType As Variant                                             *
' *                                                                     *
Public Function avSetExtension(aPath, aExt) As String
' *                                                                     *
' *  PURPOSE:  SET THE FILE EXTENSION IN A BASE NAME OR A PATH NAME    *
' *                                                                     *
' *  GIVEN:    aPath          = a base name or a full path name to be *
' *                             processed, the base name may or may   *
' *                             not contain an extension                *
' *            aExt           = extension to be set on the base name, *
' *                             should not contain a period, just the *
' *                             desired three character extension, if *
' *                             the base name has an extension, it    *
' *                             will be changed to aExt, if it does   *
' *                             not, aExt is added to the base name   *
' *                                                                     *
' *  RETURN:   avSetExtension = the new base name or full path name    *
' *                             with the specified extension applied  *
' *                                                                     *
' *  NOTE:     If aExt is a blank character (i.e. aExt = " ") or if    *
' *            the length of aExt is 0, the extension associated with *
' *            aPath will be removed, in so doing, the programmer is  *
' *            able to remove an extension from a name                  *
' *                                                                     *
' *  Dim aPath, aExt As String                                          *
' *  Dim avSetExtension As String                                       *
' *                                                                     *
Public Sub avSetExtent(pActiveView As IActiveView, _
                       pDT As IDisplayTransformation, _
                       newRect As IEnvelope)
' *                                                                     *
' *  PURPOSE:  SET THE CURRENT EXTENT OF THE VIEW                       *
' *                                                                     *
' *  GIVEN:    pActiveView = the active view                            *
' *            pDT         = the screen display transformation          *
' *            newRect     = view extent rectangle                      *
' *                                                                     *
' *  RETURN:   nothing                                                  *
' *                                                                     *
' *  Dim pActiveView As IActiveView                                     *
' *  Dim pDT As IDisplayTransformation                                  *
' *  Dim newRect As IEnvelope                                           *
' *                                                                     *
Public Sub avSetGraphicsLayer(theGLayer, pCurGraLyr As IGraphicsLayer)
' *                                                                     *
' *  PURPOSE:  TO SET THE CURRENT ANNOTATION TARGET LAYER AS THE        *
' *            BASIC GRAPHICS LAYER OR CREATE A NEW USER DEFINED        *
' *            GRAPHICS LAYER                                           *
' *                                                                     *
' *  GIVEN:    theGLayer  = graphics layer to contain the graphics      *
' *                         that are subsequently created, if NULL      *
' *                         is specified for this argument this will    *
' *                         indicate that the basic graphics layer is   *
' *                         to get the graphics that are subsequently   *
' *                         created                                     *
' *                                                                     *
' *  RETURN:   pCurGraLyr = graphics layer that will contain the user   *
' *                         programmed graphics                         *
' *                                                                     *
' *  NOTE:     (a) If theGLayer name specified exists, it will not      *
' *                be deleted, but rather, will become the current      *
' *                graphics layer, a new graphics layer will not be     *
' *                created thus any graphics generated will be added    *
' *                to the layer                                         *
' *            (b) A Map Units setting must be applied to the map in    *
' *                order for this subroutine to operate, if one is      *
' *                not, an automation error message will be generated   *
' *                                                                     *
' *  Dim theGLayer As Variant                                           *
' *  Dim pCurGraLyr As IGraphicsLayer                                   *
' *                                                                     *
Public Sub avSetName(aTitle)
' *                                                                     *
' *  PURPOSE:  TO SET THE CAPTION OF THE APPLICATION                    *
' *                                                                     *
' *  GIVEN:    aTitle  = name of the application to appear in the       *
' *                      upper left corner of the application window    *
' *                                                                     *
' *  RETURN:   nothing                                                  *
' *                                                                     *
' *  NOTE:     To set the name for a layer or table the user should     *
' *            use the subroutine avObjSetName                          *
' *                                                                     *
' *  Dim aTitle As String                                               *
' *                                                                     *
Public Sub avSetSelection(pmxDoc As IMxDocument, theTheme, _
                          psTableSel As ISelectionSet)
' *                                                                     *
' *  PURPOSE:  SET THE SELECTED SET FOR A LAYER OR TABLE                *
' *                                                                     *
' *  GIVEN:    pmxDoc     = the active view                             *
' *            theTheme   = the theme or table to be processed          *
' *            psTableSel = the selection set for the theme or table    *
' *                                                                     *
' *  RETURN:   nothing                                                  *
' *                                                                     *
' *  Dim pmxDoc As IMxDocument                                          *
' *  Dim theTheme As Variant                                            *
' *  Dim psTableSel As ISelectionSet                                    *
' *                                                                     *
Public  Sub  avSetSelectionIDs(pmxDoc  As  IMxDocument,  _
                               theTheme, selRecsList)
' *                                                                     *
' *   PURPOSE:   DEFINE A SELECTION SET GIVEN A COLLECTION OF OIDS      *
' *                                                                     *
' *  GIVEN:     pmxDoc       = the active view                          *
' *              theTheme     = theme to be processed                   *
' *               selRecsList = the list of OIDs for the selection set  *
' *                                                                     *
' *  RETURN:   nothing                                                  *
' *                                                                     *
' *  Dim pmxDoc As IMxDocument                                          *
' *  Dim theTheme As Variant                                            *
' *  Dim selRecsList as New Collection                                  *
' *                                                                     *
Public  Sub  avSetSelFeatures(pmxDoc  As  IMxDocument,  _
                              selThmList, selRecList)
' *                                                                     *
' *   PURPOSE:   TO SET THE SELECTED FEATURES FOR A SET OF THEMES       *
' *                                                                     *
' *  GIVEN:     pmxDoc      = the active view                           *
' *               selThmList = list of themes with selected features    *
' *                selRecList = list of selected features record numbers *
' *                                                                     *
' *  RETURN:   nothing                                                  *
' *                                                                     *
' *   NOTE:      (a) The records selected here will be added to the     *
' *                   current selected set for the theme                *
' *               (b) Following the call to this subroutine make a call *
' *                    to avUpdateSelection to update the display so that *
' *                   the selected features can be seen                 *
' *              (c) structure of selThmList is:                        *
' *                  Item 1: name of theme 1                            *
' *                   Item 2: number of selected features in theme 1    *
' *                  Item 3: name of theme 2                            *
' *                   Item 4: number of selected features in theme 2    *
' *                   Repeat Items 1 and 2 for each theme               *
' *              (d) structure of selRecList is:                        *
' *                   Item 1: selected feature 1 OID in theme 1         *
' *                   Item 2: selected feature 2 OID in theme 1         *
' *                    Repeat Item 1 for each selected feature in theme 1 *
' *                   Item 3: selected feature 1 OID in theme 2         *
' *                   Item 4: selected feature 2 OID in theme 2         *
' *                    Repeat Item 3 for each selected feature in theme 2 *
' *                                                                     *
' *  Dim pmxDoc As IMxDocument                                          *
' *  Dim selThmList As New Collection                                   *
' *  Dim selRecList As New Collection                                   *
' *                                                                     *
Public  Sub  avSetSelFeatures2(pmxdoc  As  IMxDocument,  _
                               selThmList, selRecList)
' *                                                                     *
' *   PURPOSE:   TO SET THE SELECTED FEATURES FOR A SET OF THEMES       *
' *               SUCH THAT THE ARCMAP EDIT TOOLS CAN PROCESS THEM      *
' *                                                                     *
' *  GIVEN:     pmxDoc      = the active view                           *
' *               selThmList = list of themes with selected features    *
' *                selRecList = list of selected features record numbers *
' *                                                                     *
' *  RETURN:   nothing                                                  *
' *                                                                     *
' *   NOTE:      (a) This subroutine is similar to avSetSelFeatures     *
' *                    with the exception that the selected features set *
' *                    by this subroutine can be manipulated by ArcMap's   *
' *                    edit tools. The selected features set with the      *
' *                    avSetSelFeatures subroutine can not be              *
' *                (b) The records selected here will be added to the      *
' *                    current selected set for the theme                  *
' *                (c) Following the call to this subroutine make a call   *
' *                    to avUpdateSelection to update the display so that *
' *                    the selected features can be seen                   *
' *                (d) structure of selThmList is:                         *
' *                    Item 1: name of theme 1                             *
' *                    Item 2: number of selected features in theme 1      *
' *                    Item 3: name of theme 2                             *
' *                    Item 4: number of selected features in theme 2      *
' *                    Repeat Items 1 and 2 for each theme                 *
' *                (e) structure of selRecList is:                         *
' *                    Item 1: selected feature 1 OID in theme 1           *
' *                    Item 2: selected feature 2 OID in theme 1           *
' *                    Repeat Item 1 for each selected feature in theme 1 *
' *                    Item 3: selected feature 1 OID in theme 2           *
' *                    Item 4: selected feature 2 OID in theme 2           *
' *                    Repeat Item 3 for each selected feature in theme 2 *
' *                                                                        *
' *  Dim pmxDoc As IMxDocument                                             *
' *  Dim selThmList As New Collection                                      *
' *  Dim selRecList As New Collection                                      *
' *                                                                        *
```

```
Public Sub avSetValue(pmxDoc As IMxDocument, theTheme, _
                      aField, aRecord, anObj)
```

```
' *                                                                        *
' *  PURPOSE:   TO STORE A VALUE IN A SPECIFIC FIELD OF A SPECIFIC         *
' *             ROW (RECORD) FOR A LAYER OR TABLE                          *
' *                                                                        *
' *  GIVEN:     pmxDoc   = the active view                                 *
' *             theTheme = the theme or table to be processed              *
' *             aField   = index value denoting field to be written to    *
' *             aRecord  = record of theme or table to be processed        *
' *             anObj    = object to be stored (not geometry but only      *
' *                        attribute information, see note a below)        *
' *                        set this value to be the string, StoreRec,      *
' *                        when the record, aRecord, is to be written      *
' *                        to disk, see note b below                       *
' *                                                                        *
' *  RETURN:    nothing                                                    *
' *                                                                        *
' *  NOTE:      (a) Do not use this routine to store geometry in the       *
' *                 SHAPE field, use this routine to store attribute       *
' *                 information only.  Use avSetValueG to store            *
' *                 geometry in the SHAPE field                            *
' *             (b) This procedure does not write the record, aRecord,    *
' *                 to disk until the procedure is called with the         *
' *                 argument, anObj, set to "StoreRec".  This is done      *
' *                 to eliminate multiple disk writes thereby yielding    *
' *                 increased performance                                  *
' *             (c) When the argument, anObj, is set to "StoreRec",        *
' *                 the argument, aField, is ignored                       *
' *                                                                        *
' *  Dim pmxDoc As IMxDocument                                             *
' *  Dim theTheme As Variant                                               *
' *  Dim aField, aRecord As Long                                           *
' *  Dim anObj As Variant                                                  *
' *                                                                        *
```

```
Public Sub avSetValueG(pmxDoc As IMxDocument, theTheme, _
                       aField, aRecord, aShape As IGeometry)
```

```
' *                                                                        *
' *  PURPOSE:   TO STORE A SHAPE IN THE SHAPE FIELD OF A SPECIFIC ROW      *
' *             FOR A LAYER                                                *
' *                                                                        *
' *  GIVEN:     pmxDoc   = the active view                                 *
' *             theTheme = the theme to be processed                       *
' *              aField    = the shape field (not used but included only *
' *                          for compatibility purposes)                 *
' *              aRecord  = record of theme or table to be processed     *
' *              aShape    = shape to be stored (not attribute data but  *
' *                          only geometry, see note a below)            *
' *                                                                      *
' *  RETURN:   nothing                                                   *
' *                                                                      *
' *   NOTE:      (a) Do not use this routine to store attribute data,    *
' *                   use this routine to store geometry only.  Use the  *
' *                   routine avSetValue to store attribute data.        *
' *               (b) This procedure will write the record to disk after *
' *                  the shape has been stored.                          *
' *               (c) Calling this procedure after calling avSetValue    *
' *                   eliminates the need to call avSetValue with the    *
' *                   anObj argument set to "StoreRec" because this      *
' *                  procedure writes the record to disk                 *
' *                                                                      *
' *  Dim pmxDoc As IMxDocument                                           *
' *  Dim theTheme As Variant                                             *
' *  Dim aField, aRecord As Long                                         *
' *  Dim aShape As IGeometry                                             *
' *                                                                      *
Public  Sub  avSetVisible(name,  aStatus)
' *                                                                      *
' *   PURPOSE:  TO SET THE VISIBILITY STATUS OF AN OBJECT                *
' *                                                                      *
' *   GIVEN:     name     = name of input object for which its           *
' *                          visibility status is to be defined          *
' *              aStatus = the visible state of the input object         *
' *                          true = visible, false = not visible         *
' *                                                                      *
' *  RETURN:   nothing                                                   *
' *                                                                      *
' *  Dim name As Variant                                                 *
' *  Dim aStatus As Boolean                                              *
' *                                                                      *
Public  Sub  avSetWorkDir(theWorkDir)
' *                                                                      *
' *   PURPOSE:  SET THE CURRENT WORKING DIRECTORY                        *
' *                                                                      *
' *   GIVEN:     theWorkDir = the new working directory                  *
' *                                                                      *
' *  RETURN:   nothing                                                   *
' *                                                                      *
' *  Dim theWorkDir As String                                            *
' *                                                                      *
Public  Sub  avShowMsg(aMessage)
' *                                                                      *
' *   PURPOSE:  DISPLAY A MESSAGE IN THE STATUS BAR AREA                 *
' *                                                                      *
' *   GIVEN:     aMessage = the message to be displayed                  *
' *                                                                      *
' *  RETURN:   nothing                                                   *
' *                                                                      *
' *  Dim aMessage As String                                              *
' *                                                                      *
Public  Sub  avShowStopButton()
' *                                                                      *
' *   PURPOSE:  DISPLAY THE STOP BUTTON ON THE PROGRESS BAR              *
' *                                                                      *
' *  GIVEN:     nothing                                                  *
' *                                                                      *
' *  RETURN:   nothing                                                   *
' *                                                                      *
' *   NOTE:      Use of this command will result in the progress bar     *
' *               appearing in the middle of the display and not in the  *
' *              status bar area                                         *
' *                                                                      *
```

```
Public  Sub  avSingleSymbol(pmxDoc  As  IMxDocument,  _
                                theTheme, pDesc, pLabel, pSym As ISymbol)
' *                                                                      *
' *   PURPOSE:   TO SET THE LEGEND THAT IS ASSOCIATED WITH A THEME TO    *
' *              BE OF SINGLE SYMBOL TYPE                                *
' *                                                                      *
' *  GIVEN:     pmxDoc    = the active view                              *
' *              theTheme = theme to be processed                        *
' *             pDesc     = renderer description                         *
' *               pLabel    = label (appears in the Table of Contents)   *
' *               pSym      = symbol used to draw every feature in theme *
' *                                                                      *
' *  RETURN:    nothing                                                  *
' *                                                                      *
' *   NOTE:      (a) All features in the theme will be classified such   *
' *                   that every feature is drawn using the same         *
' *                 symbology:  color, etc.                              *
' *               (b) pDesc and pLabel can be specified as NULL and pSym *
' *                   as NOTHING if default values are to be used for    *
' *                 these parameters                                     *
' *               (c) After a theme has been classified you must use     *
' *                   avGetLegend to get the new legend that reflects    *
' *                   the new classification if you wish to manipulate   *
' *                   the labels or symbols in the classification        *
' *                                                                      *
' *  Dim pmxDoc As IMxDocument                                           *
' *  Dim theTheme As Variant                                             *
' *  Dim pDesc As String                                                 *
' *  Dim pLabel As String                                                *
' *  Dim pSym As ISymbol                                                 *
' *                                                                      *
Public  Sub  avSplit(aShape1  As  IGeometry,  _
                          aShape2 As IGeometry, shapeList)
' *                                                                      *
' *   PURPOSE:   TO SPLIT A SHAPE USING A SECOND SHAPE AS THE SPLITTER   *
' *                                                                      *
' *  GIVEN:     aShape1    = shape to be split                           *
' *               aShape2    = shape to be used as the split line        *
' *                                                                      *
' *   RETURN:    shapeList  =  list of new shapes created as a result of *
' *                              the splitting process                   *
' *                                                                      *
' *  Dim aShape1 As IGeometry                                            *
' *  Dim aShape2 As IGeometry                                            *
' *  Dim shapeList As New Collection                                     *
' *                                                                      *
Public  Sub  avStartOperation()
' *                                                                      *
' *   PURPOSE:  TO START AN OPERATION WITHIN AN EDIT SESSION             *
' *                                                                      *
' *  GIVEN:     nothing                                                  *
' *                                                                      *
' *  RETURN:    nothing                                                  *
' *                                                                      *
' *   NOTE:      (a) The global variable ugEditMode is used to keep      *
' *                   track of if an operation is or is not in progress  *
' *                   If the value of ugEditMode is 0, an operation has  *
' *                   not been started, if ugEditMode is 1, an operation *
' *                  has been started. A new operation can not be        *
' *                  started if one is currently in progress             *
' *               (b) The theme or table must be editable prior to using *
' *                 this subroutine                                      *
' *               (c) If an error occurs during the processing of this   *
' *                   subroutine, no error message will be generated,    *
' *                   but rather, the subroutine will simply gracefully  *
' *                   terminate without displaying an error message      *
' *                                                                      *
```

```
Public  Sub  avStopEditing()
' *                                                                       *
' *   PURPOSE:   TERMINATE THE EDITING ON ALL LAYERS AND TABLES           *
' *                                                                       *
' *  GIVEN:     nothing                                                   *
' *                                                                       *
' *  RETURN:    nothing                                                   *
' *                                                                       *
' *   NOTE:       (a) This command when used will empty the Undo list so *
' *                    that the user will not be able to use the Edit sub *
' *                    menu item Undo (all edits are committed to disk)   *
' *               (b) If the editor is not in an edit state, an error     *
' *                    message will not be generated, but rather, no      *
' *                  action will take place                               *
' *                                                                       *
' *  Dim pmxDoc As IMxDocument                                            *
' *  Dim theTheme As Variant                                              *
' *                                                                       *
Public  Sub  avStopOperation(oprMssg)
' *                                                                       *
' *   PURPOSE:   TO STOP AN OPERATION WITHIN AN EDIT SESSION              *
' *                                                                       *
' *   GIVEN:      oprMssg = edit operation message that will appear to    *
' *                          the right of the Undo menu item under the    *
' *                        Edit menu item                                 *
' *                                                                       *
' *  RETURN:    nothing                                                   *
' *                                                                       *
' *   NOTE:       This subroutine does not stop the editor, it will only *
' *                terminate an operation. When the editor is stopped, it *
' *                is not possible to use the Undo command under the Edit *
' *                menu item, so that, if the Undo command is to be used, *
' *              the Editor must be active (in use)                       *
' *                                                                       *
' *  Dim oprMssg As Variant                                               *
' *                                                                       *
Public  Function  avSummarize(pmxDoc  As  IMxDocument,  theTheme,  _
                              aFileName, aType, aField, _
                              fieldList, sumryList) As ITable
' *                                                                       *
' *   PURPOSE:   TO SUMMARIZE A THEME OR A TABLE ON A SPECIFIC FIELD      *
' *               THE RECORDS PROCESSED ARE THOSE THAT ARE SELECTED       *
' *                                                                       *
' *  GIVEN:     pmxDoc       = the active view                            *
' *               theTheme     = theme or table to be processed           *
' *                aFileName    = name of the output table to be created, *
' *                               table will be stored in the workspace of *
' *                               the theme or table that is summarized so *
' *                              do not specify a full pathname and do    *
' *                              not include an extension such as .dbf    *
' *                              If an extension appears in the name it   *
' *                               will be removed with no error generated *
' *              aType        = type of output table                      *
' *                             "dBase"                                   *
' *                aField       = field that theme or table summarized on *
' *               fieldList    = list of fields to be summarized          *
' *                sumryList    = type of summary to be performed on items *
' *                              in the fieldList (operation codes)       *
' *                              Dissolve (for use on the Shape field)    *
' *                             Count                                     *
' *                             Minimum                                   *
' *                             Maximum                                   *
' *                             Sum                                       *
' *                             Average                                   *
' *                             Variance                                  *
' *                             StdDev                                    *
' *                                                                       *
' *   RETURN:    avSummarize  =  list of attributes in summarized table,  *
' *                              will be set to NOTHING if an error was   *
' *                              encountered during the processing         *
' *                                                                        *
' *   NOTE:       (a) Since this routine passes avSummarize as NOTHING     *
' *                    if an error is detected, make sure to check for     *
' *                    this in the code that calls this function           *
' *               (b) If fieldList and sumryList are empty lists or        *
' *                    passed in as NOTHING default values will be used,   *
' *                    that is, Count.aField and Maximum.aField            *
' *               (c) If the table to be created exists on disk, the       *
' *                    routine will overwrite the existing table without   *
' *                   asking or informing the user                         *
' *               (d) If the table contains selected records, then only    *
' *                    the selected records will be processed, if there    *
' *                    are no selected records, then the entire table      *
' *                   will be processed                                    *
' *                                                                        *
' *  Dim pmxDoc As IMxDocument                                             *
' *  Dim theTheme As Variant                                               *
' *  Dim aFileName, aType, aField As String                                *
' *  Dim fieldList, sumryList As New Collection                            *
' *  Dim avSummarize As ITable                                             *
' *                                                                        *
```
Public Function avSymbolGetAngle(aSymTyp, pSymbol As ISymbol) As Double
```
' *                                                                        *
' *   PURPOSE:   TO GET THE ANGLE ASSIGNED TO A GRAPHIC SYMBOL             *
' *                                                                        *
' *   GIVEN:     aSymTyp            = type of symbol to be processed       *
' *                                   PEN    : line symbol                 *
' *                                   MARKER : point symbol                *
' *                                   FILL   : polygon symbol              *
' *              pSymbol            = symbol to be processed               *
' *                                                                        *
' *   RETURN:     avSymbolGetAngle = angle assigned to symbol (degrees)    *
' *                                                                        *
' *   NOTE:       (a) This routine processes only MARKER symbols, the      *
' *                    PEN and FILL symbols will result in a value of      *
' *                   zero for avSymbolGetAngle                            *
' *               (b) For text symbols use avGraphicTextGetAngle to get    *
' *                   the text angle                                       *
' *                                                                        *
' *  Dim aSymTyp As String                                                 *
' *  Dim pSymbol As ISymbol                                                *
' *  Dim avSymbolGetAngle As Double                                        *
' *                                                                        *
```
Public Function avSymbolGetColor(aSymTyp, pSymbol As ISymbol) As iColor
```
' *                                                                        *
' *   PURPOSE:   TO GET THE COLOR ASSIGNED TO A GRAPHIC SYMBOL             *
' *                                                                        *
' *   GIVEN:     aSymTyp            = type of symbol to be processed       *
' *                                   PEN    : line symbol                 *
' *                                   MARKER : point symbol                *
' *                                   FILL   : polygon symbol              *
' *                                   TEXT   : text symbol                 *
' *              pSymbol            = symbol to be processed               *
' *                                                                        *
' *   RETURN:    avSymbolGetColor = color assigned to symbol               *
' *                                                                        *
' *   NOTE:       It is possible for avSymbolGetColor to be NOTHING so     *
' *                make sure to check for this condition before using the *
' *                result (i.e. some polygon fills have no color, so that *
' *                avSymbolGetColor will be NOTHING in those instances)    *
' *                                                                        *
' *  Dim aSymTyp As String                                                 *
' *  Dim pSymbol As ISymbol                                                *
' *  Dim avSymbolGetColor As iColor                                        *
' *                                                                        *
```

```
Public  Function  avSymbolGetOLColor(aSymTyp,  _
                             pSymbol As ISymbol) As iColor
' *                                                                       *
' *   PURPOSE:   TO GET THE OUTLINE COLOR ASSIGNED TO A GRAPHIC SYMBOL    *
' *                                                                       *
' *  GIVEN:     aSymTyp              = type of symbol to be processed     *
' *                                    PEN     : line symbol              *
' *                                    MARKER : point symbol              *
' *                                    FILL    : polygon symbol           *
' *             pSymbol              = symbol to be processed             *
' *                                                                       *
' *   RETURN:    avSymbolGetOLColor = color assigned to symbol            *
' *                                                                       *
' *   NOTE:       It is possible for avSymbolGetOLColor to be NOTHING so *
' *               make sure to check for this condition before using the *
' *               result (i.e. some polygon fills have no color, so that *
' *               avSymbolGetOLColor will be NOTHING in those instances) *
' *                                                                       *
' *  Dim aSymTyp As String                                                *
' *  Dim pSymbol As ISymbol                                               *
' *  Dim avSymbolGetOLColor As iColor                                     *
' *                                                                       *
Public  Function  avSymbolGetOLWidth(aSymTyp,  _
                                       pSymbol As ISymbol) As Double
' *                                                                       *
' *   PURPOSE:   TO GET THE OUTLINE WIDTH ASSIGNED TO A GRAPHIC SYMBOL    *
' *                                                                       *
' *  GIVEN:     aSymTyp              = type of symbol to be processed     *
' *                                    PEN     : line symbol              *
' *                                    MARKER : point symbol              *
' *                                    FILL    : polygon symbol           *
' *             pSymbol              = symbol to be processed             *
' *                                                                       *
' *   RETURN:     avSymbolGetOLWidth = outline width assigned to symbol   *
' *                                                                       *
' *   NOTE:       For PEN symbols the width of the symbol is assigned to *
' *                avSymbolGetOLWidth, for MARKER symbols if the outline  *
' *               is to be drawn the outline size is returned, otherwise *
' *             the size of the marker will be returned                  *
' *                                                                       *
' *  Dim aSymTyp As String                                                *
' *  Dim pSymbol As ISymbol                                               *
' *  Dim avSymbolGetOLWidth As Double                                     *
' *                                                                       *
Public  Function  avSymbolGetSize(aSymTyp,  pSymbol  As  ISymbol)  As  Double
' *                                                                       *
' *   PURPOSE:   TO GET THE SIZE ASSIGNED TO A GRAPHIC SYMBOL             *
' *                                                                       *
' *  GIVEN:     aSymTyp           = type of symbol to be processed        *
' *                                 PEN     : line symbol                 *
' *                                 MARKER : point symbol                 *
' *                                 FILL    : polygon symbol              *
' *                                 TEXT    : text symbol                 *
' *             pSymbol           = symbol to be processed                *
' *                                                                       *
' *   RETURN:    avSymbolGetSize = size assigned to symbol                *
' *                                                                       *
' *   NOTE:      For PEN and FILL symbols the width of the symbol is     *
' *              assigned to avSymbolGetSize                              *
' *                                                                       *
' *  Dim aSymTyp As String                                                *
' *  Dim pSymbol As ISymbol                                               *
' *  Dim avSymbolGetSize As Double                                        *
' *                                                                       *
Public  Function  avSymbolGetStipple(aSymTyp,  _
                                 pSymbol As ISymbol) As IMultiLayerFillSymbol
' *                                                                       *
' *   PURPOSE:   TO GET THE STIPPLE ASSIGNED TO A GRAPHIC SYMBOL          *
' *                                                                       *
' *  GIVEN:     aSymTyp              = type of symbol to be processed     *
' *                                    PEN    : line symbol               *
' *                                    MARKER : point symbol              *
' *                                    FILL   : polygon symbol            *
' *             pSymbol              = symbol to be processed             *
' *                                                                       *
' *  RETURN:    avSymbolGetStipple  =  IMultiLayerFillSymbol interface    *
' *                                    for the symbol if it is of this    *
' *                                    type, otherwise, NOTHING           *
' *                                                                       *
' *  NOTE:      Since there is no direct correlation between ArcView      *
' *             .Stipple request and an ArcObject method or property,     *
' *             we will use this macro to return an object of type        *
' *             IMultiLayerFillSymbol provided the symbol is of that      *
' *             type, otherwise, NOTHING will be passed back              *
' *                                                                       *
' *  Dim aSymTyp As String                                                *
' *  Dim pSymbol As ISymbol                                               *
' *  Dim avSymbolGetStipple As IMultiLayerFillSymbol                      *
' *                                                                       *
Public  Function  avSymbolGetStyle(aSymTyp,  _
                                   pSymbol As ISymbol) As Variant
' *                                                                       *
' *  PURPOSE:  TO GET THE STYLE ASSIGNED TO A GRAPHIC SYMBOL              *
' *                                                                       *
' *  GIVEN:     aSymTyp            = type of symbol to be processed       *
' *                                  PEN    : line symbol                 *
' *                                  MARKER : point symbol                *
' *                                  FILL   : polygon symbol              *
' *             pSymbol            = symbol to be processed               *
' *                                                                       *
' *  RETURN:    avSymbolGetStyle = style assigned to symbol,              *
' *                                  varies depending upon symbol type    *
' *                                  for PEN symbols                      *
' *                                  0 : Solid                            *
' *                                  1 : Dashed                           *
' *                                  2 : Dotted                           *
' *                                  3 : dashes & dots                    *
' *                                  4 : dashes & double dots             *
' *                                  5 : Is invisible                     *
' *                                  6 : Fit into bounding rectangle      *
' *                                  for MARKER symbols                   *
' *                                  0 : Circle                           *
' *                                  1 : Square                           *
' *                                  2 : Cross                            *
' *                                  3 : X                                *
' *                                  4 : Diamond                          *
' *                                  for FILL symbols                     *
' *                                  0 : Solid                            *
' *                                  1 : Empty                            *
' *                                  2 : Horizontal hatch                 *
' *                                  3 : Vertical hatch                   *
' *                                  4 : 450 left-to-right hatch          *
' *                                  5 : 450 left-to-right hatch          *
' *                                  6 : Horz. and vert. crosshatch       *
' *                                  7 : 450 crosshatch                   *
' *                                                                       *
' *  NOTE:      This routine processes only ISimpleLineSymbol,            *
' *             ISimpleMarkerSymbol and ISimpleFillSymbol type symbols    *
' *                                                                       *
' *  Dim aSymTyp As String                                                *
' *  Dim pSymbol As ISymbol                                               *
' *  Dim avSymbolGetStyle As Variant                                      *
' *                                                                       *
Public  Function  avSymbolMake(aSymTyp)  As  ISymbol
' *                                                                       *
' *  PURPOSE:  TO CREATE A NEW GRAPHIC SYMBOL                             *
' *                                                                       *
' *  GIVEN:     aSymTyp       = type of graphic to be created             *
' *                              PEN     : line symbol                    *
' *                              MARKER : point symbol                    *
' *                              FILL    : polygon symbol                 *
' *                                                                       *
' *   RETURN:    avSymbolMake  =  symbol describing a graphic that can be *
' *                              added to the graphics layer              *
' *                                                                       *
' *   NOTE:      (a) This routine will create only ISimpleLineSymbol,     *
' *                   ISimpleMarkerSymbol and ISimpleFillSymbol type      *
' *                 symbols                                               *
' *              (b) For text symbols use MakeTextSymbol to create a      *
' *                   text symbol or avGraphicTextMake to create a text   *
' *                 element                                               *
' *                                                                       *
' *  Dim aSymTyp As String                                                *
' *  Dim avSymbolMake As ISymbol                                          *
' *                                                                       *
Public  Sub  avSymbolSetAngle(aSymTyp,  pSymbol  As  ISymbol,  aAngle)
' *                                                                       *
' *  PURPOSE:  TO SET THE ANGLE OF A SYMBOL                               *
' *                                                                       *
' *  GIVEN:     aSymTyp = type of symbol to be processed                  *
' *                        PEN     : line symbol                          *
' *                        MARKER : point symbol                          *
' *                        FILL    : polygon symbol                       *
' *              pSymbol = symbol to be processed                         *
' *              aAngle  = angle to be assigned (degrees)                 *
' *                                                                       *
' *  RETURN:    nothing                                                   *
' *                                                                       *
' *   NOTE:      For text symbols use avGraphicTextSetAngle to define     *
' *             the text angle                                            *
' *                                                                       *
' *  Dim aSymTyp As String                                                *
' *  Dim pSymbol As ISymbol                                               *
' *  Dim aAngle As Variant                                                *
' *                                                                       *
Public  Sub  avSymbolSetColor(aSymTyp,  pSymbol  As  ISymbol,  aColor)
' *                                                                       *
' *  PURPOSE:  TO SET THE COLOR FOR A SYMBOL                              *
' *                                                                       *
' *  GIVEN:     aSymTyp = type of symbol to be processed                  *
' *                        PEN     : line symbol                          *
' *                        MARKER : point symbol                          *
' *                        FILL    : polygon symbol                       *
' *                        TEXT    : text symbol                          *
' *              pSymbol = symbol to be processed                         *
' *               aColor   = color to be assigned, if numeric will refer  *
' *                          to a RGB color index value, otherwise one of *
' *                          the predefined values listed below           *
' *                       WHITE                                           *
' *                       BLACK                                           *
' *                       BLUE                                            *
' *                       GREEN                                           *
' *                       YELLOW                                          *
' *                       CYAN                                            *
' *                       BROWN                                           *
' *                       ORANGE                                          *
' *                       RED                                             *
' *                       MAGENTA                                         *
' *                       GRAY                                            *
' *                       LIGHT GRAY                                      *
' *                                                                       *
' *  RETURN:    nothing                                                   *
' *                                                                       *
' *  Dim aSymTyp As String                                                *
' *  Dim pSymbol As ISymbol                                               *
' *  Dim aColor As Variant                                                *
' *                                                                       *
```

```
Public  Sub  avSymbolSetOLColor(aSymTyp,  pSymbol  As  ISymbol,  aColor)
' *                                                                    *
' *  PURPOSE:  TO SET THE OUTLINE COLOR FOR A SYMBOL                   *
' *                                                                    *
' *  GIVEN:     aSymTyp = type of symbol to be processed               *
' *                        PEN    : line symbol                        *
' *                        MARKER : point symbol                       *
' *                        FILL   : polygon symbol                     *
' *             pSymbol = symbol to be processed                       *
' *              aColor  = color to be assigned, if numeric will refer *
' *                         to a RGB color index value, otherwise one of *
' *                        the predefined values listed below          *
' *                       WHITE                                        *
' *                       BLACK                                        *
' *                       BLUE                                         *
' *                       GREEN                                        *
' *                       YELLOW                                       *
' *                       CYAN                                         *
' *                       BROWN                                        *
' *                       ORANGE                                       *
' *                       RED                                          *
' *                       MAGENTA                                      *
' *                       GRAY                                         *
' *                       LIGHT GRAY                                   *
' *                                                                    *
' *  RETURN:   nothing                                                 *
' *                                                                    *
' *  NOTE:      For PEN symbols the color of the symbol is set to the  *
' *              value of aColor, for MARKER symbols the outline       *
' *              property is set to be true, denoting that the outline *
' *              for the marker is to be drawn, and the outline color  *
' *             is set to the value of aColor                          *
' *                                                                    *
' *  Dim aSymTyp As String                                             *
' *  Dim pSymbol As ISymbol                                            *
' *  Dim aColor As Variant                                             *
' *                                                                    *
Public  Sub  avSymbolSetOLWidth(aSymTyp,  pSymbol  As  ISymbol,  aWidth)
' *                                                                    *
' *  PURPOSE:  TO SET THE OUTLINE WIDTH FOR A SYMBOL                   *
' *                                                                    *
' *  GIVEN:     aSymTyp = type of symbol to be processed               *
' *                        PEN    : line symbol                        *
' *                        MARKER : point symbol                       *
' *                        FILL   : polygon symbol                     *
' *             pSymbol = symbol to be processed                       *
' *              aWidth  = outline width to be assigned, a value of    *
' *                        zero denotes no outline is to be drawn      *
' *                                                                    *
' *  RETURN:   nothing                                                 *
' *                                                                    *
' *  NOTE:      For PEN symbols the width of the symbol is set to the  *
' *              value of aWidth, for MARKER symbols the outline       *
' *              property is set to be true, denoting that the outline *
' *              for the marker is to be drawn, and the outline width  *
' *             is set to the value of aWidth                          *
' *                                                                    *
' *  Dim aSymTyp As String                                             *
' *  Dim pSymbol As ISymbol                                            *
' *  Dim aWidth As Variant                                             *
' *                                                                    *
Public  Sub  avSymbolSetSize(aSymTyp,  pSymbol  As  ISymbol,  aSize)
' *                                                                    *
' *  PURPOSE:  TO SET THE SIZE OF A SYMBOL                             *
' *                                                                    *
' *  GIVEN:     aSymTyp = type of symbol to be processed               *
' *                        PEN    : line symbol                        *
' *                        MARKER : point symbol                       *
' *                        FILL   : polygon symbol                     *
' *                         TEXT    : text symbol                          *
' *             pSymbol = symbol to be processed                           *
' *              aSize    = size to be assigned (greater than zero)        *
' *                                                                        *
' *  RETURN:   nothing                                                     *
' *                                                                        *
' *   NOTE:      For PEN and FILL symbols this routine works the same      *
' *             as avSymbolSetOLWidth                                      *
' *                                                                        *
' *  Dim aSymTyp As String                                                 *
' *  Dim pSymbol As ISymbol                                                *
' *  Dim aSize As Variant                                                  *
' *                                                                        *
Public  Sub  avSymbolSetStipple(aSymTyp,  pSymbol  As  ISymbol,  _
                               aStipple As IMultiLayerFillSymbol)
' *                                                                        *
' *  PURPOSE:  TO SET THE STIPPLE OF A SYMBOL                              *
' *                                                                        *
' *  GIVEN:     aSymTyp  = type of symbol to be processed                  *
' *                         PEN     : line symbol                          *
' *                         MARKER : point symbol                          *
' *                         FILL    : polygon symbol                       *
' *             pSymbol  = symbol to be processed                          *
' *             aStipple = stipple to be assigned                          *
' *                                                                        *
' *  RETURN:   nothing                                                     *
' *                                                                        *
' *   NOTE:      Since there is no direct correlation between ArcView      *
' *               .Stipple request and an ArcObject method or property,    *
' *              we will use this macro to allow the user to change a      *
' *               ISymbol object into a IMultiLayerFillSymbol provided     *
' *              a valid IMultiLayerFillSymbol object is given             *
' *                                                                        *
' *  Dim aSymTyp As String                                                 *
' *  Dim pSymbol As ISymbol                                                *
' *  Dim aStipple As IMultiLayerFillSymbol                                 *
' *                                                                        *
Public  Sub  avSymbolSetStyle(aSymTyp,  pSymbol  As  ISymbol,  aStyle)
' *                                                                        *
' *  PURPOSE:  TO SET THE STYLE FOR A SYMBOL                               *
' *                                                                        *
' *  GIVEN:     aSymTyp = type of symbol to be processed                   *
' *                        PEN     : line symbol                           *
' *                        MARKER : point symbol                           *
' *                        FILL    : polygon symbol                        *
' *             pSymbol = symbol to be processed                           *
' *               aStyle  = style to be assigned, varies depending upon    *
' *                        the type of symbol                              *
' *                        for PEN symbols                                 *
' *                        0 : Solid                                       *
' *                        1 : Dashed                                      *
' *                        2 : Dotted                                      *
' *                         3 : Has alternating dashes and dots            *
' *                          4 : Has alternating dashes and double dots    *
' *                        5 : Is invisible                                *
' *                         6 : Will fit into it's bounding rectangle      *
' *                        for MARKER symbols                              *
' *                        0 : Circle                                      *
' *                        1 : Square                                      *
' *                        2 : Cross                                       *
' *                        3 : X                                           *
' *                        4 : Diamond                                     *
' *                        for FILL symbols                                *
' *                        0 : Solid                                       *
' *                        1 : Empty                                       *
' *                        2 : Horizontal hatch                            *
' *                        3 : Vertical hatch                              *
' *                          4 : 45-degree downward, left-to-right hatch   *
' *                          5 : 45-degree upward, left-to-right hatch     *
' *                          6 : Horizontal and vertical crosshatch        *
' *                         7 : 45-degree crosshatch                        *
' *                                                                         *
' *  RETURN:   nothing                                                      *
' *                                                                         *
' *  Dim aSymTyp As String                                                  *
' *  Dim pSymbol As ISymbol                                                 *
' *  Dim aStyle As Variant                                                  *
' *                                                                         *
Public  Sub  avTableSort(tblName,  aField,  anOrder,  Optional  aFileName)
' *                                                                          *
' *   PURPOSE:   TO SORT A TABLE BASED UPON A FIELD IN AN ASCENDING OR   *
' *              DESCENDING ORDER                                           *
' *                                                                         *
' *  GIVEN:     tblName    = name of table to be sorted                     *
' *              aField     = name of field that the sort is based upon   *
' *             anOrder    = the sort order as a Boolean                    *
' *                          True = ascending, False = Descending           *
' *               aFileName = optional argument denoting the name of the *
' *                          new dBase file that will be created            *
' *                          if the name does not contain a complete        *
' *                          pathname the current working directory         *
' *                          will be used, some examples include:           *
' *                                 c:\project\test\atable.dbf              *
' *                                atable.dbf                               *
' *                                                                         *
' *  RETURN:   nothing                                                      *
' *                                                                         *
' *   NOTE:       (a) If the table contains selected records then only     *
' *                    the selected records will be sorted, otherwise,      *
' *                   the entire table will be sorted                       *
' *               (b) A new dBase file is created containing the results *
' *                   of the sort and is added to the document, the        *
' *                    default name of the new dBase file will be of the  *
' *                  form tblName_sort.dbf                                  *
' *               (c) The optional argument, aFileName, can be used to     *
' *                    explicitly define the name of the new dBase file    *
' *                   and override the default naming convention           *
' *               (d) If the new dBase file that is to be created exists *
' *                    on disk, it will be deleted prior to creating the  *
' *                  new file                                               *
' *               (e) If the new dBase table that is to be added to the   *
' *                    current document exists in the document, it will    *
' *                   be removed prior to adding it back in                 *
' *                                                                         *
' *  Dim tblName, aField As String                                          *
' *  Dim anOrder As Boolean                                                 *
' *  Dim aFileName As String                                                *
' *                                                                         *
Public  Sub  avThemeInvalidate(pmxDoc  As  IMxDocument,  theTheme,  rdStatus)
' *                                                                         *
' *  PURPOSE:  REDRAW A THEME                                               *
' *                                                                         *
' *  GIVEN:     pmxDoc    = the active view                                 *
' *              theTheme = theme to be processed                           *
' *               rdStatus = redraw status (True = redraw entire view)    *
' *                                        (False = redraw theme only)    *
' *                                                                         *
' *  RETURN:   nothing                                                      *
' *                                                                         *
' *   NOTE:       If False is specified for rdStatus it may be necessary *
' *               to follow the call to avThemeInvalidate with a call to *
' *                pActiveView.Refresh or avGetDisplayFlush in order to    *
' *               refresh the display so that the changes made to the      *
' *                theme are properly displayed, these calls are not made *
' *               here thereby eliminating multiple screen redraws         *
' *                                                                         *
' *  Dim pmxDoc As IMxDocument                                              *
' *  Dim theTheme As Variant                                                *
' *  Dim rdStatus As Boolean                                                *
' *                                                                         *
Public  Sub  avThemeSetName(pmxDoc  As  IMxDocument,  _
                              theTheme, newName, updateTOC)
' *                                                                         *
' *  PURPOSE:  TO SET THE NAME OR ALIAS FOR A LAYER                         *
' *                                                                         *
' *  GIVEN:    pmxDoc    = the active view                                  *
' *            theTheme  = theme to be processed                            *
' *            newName   = new name or alias to be assigned to theme        *
' *            updateTOC = update status (True = update TOC)                *
' *                                     (False = do not update TOC)         *
' *                                                                         *
' *  RETURN:   nothing                                                      *
' *                                                                         *
' *  NOTE:     The given updateTOC allows the user to control when          *
' *            the table of contents (TOC) is refreshed, to reflect         *
' *            the name change.  If many layers are to be modified          *
' *            it is better to update at the end of the modifications       *
' *            rather than after every single modification                  *
' *                                                                         *
' *  Dim pmxDoc As IMxDocument                                              *
' *  Dim theTheme As Variant                                                *
' *  Dim newName As Variant                                                 *
' *  Dim updateTOC As Boolean                                               *
' *                                                                         *
Public  Sub  avUnion(pMap  As  IMap,  geomList,  pNewGeom  As  IGeometry)
' *                                                                         *
' *  PURPOSE:  TO UNION SHAPES OF THE SAME GEOMETRY TYPE INTO ONE           *
' *            NEW SHAPE                                                    *
' *                                                                         *
' *  GIVEN:    pMap     = IMap object for current active view               *
' *            geomList = list of geometry objects to be unioned            *
' *                                                                         *
' *  RETURN:   pNewGeom = new geometry object representing the union        *
' *                       of the given geometry objects                     *
' *                                                                         *
' *  Dim pMap As IMap                                                       *
' *  Dim geomList As New Collection                                         *
' *  Dim pNewGeom As IGeometry                                              *
' *                                                                         *
Public  Sub  avUnique(pmxDoc  As  IMxDocument,  theTheme,  aField,  showNulls)
' *                                                                         *
' *  PURPOSE:  TO SET THE LEGEND THAT IS ASSOCIATED WITH A THEME            *
' *            TO BE OF UNIQUE TYPE                                         *
' *                                                                         *
' *  GIVEN:    pmxDoc    = the active view                                  *
' *            theTheme  = theme to be processed                            *
' *            aField    = field name that theme is to be classified        *
' *                        upon                                             *
' *            showNulls = flag denoting whether features that have         *
' *                        not been assigned a value for aField             *
' *                        should be drawn or not (true, false)             *
' *                                                                         *
' *  RETURN:   nothing                                                      *
' *                                                                         *
' *  NOTE:     (a) All features in the theme will be classified such        *
' *                that features having a unique value, within a            *
' *                field, will be drawn in a unique or different            *
' *                symbol from the other unique values within the           *
' *                field                                                    *
' *            (b) After a theme has been classified you must use           *
' *                avGetLegend to get the new legend that reflects          *
' *                the new classification if you wish to manipulate         *
' *                the labels or symbols in the classification              *
' *                                                                         *
' *  Dim pmxDoc As IMxDocument                                              *
' *  Dim theTheme As Variant                                                *
' *  Dim aField As String                                                   *
' *  Dim showNulls As Boolean                                             *
' *                                                                      *
Public  Sub  avUniqueM(pmxDoc  As  IMxDocument, _
                           theTheme, aField1, aField2, aField3, showNulls)
' *                                                                      *
' *   PURPOSE:   TO SET THE LEGEND THAT IS ASSOCIATED WITH A THEME       *
' *                TO BE OF UNIQUE TYPE FOR MULTIPLE ATTRIBUTES (MAX 3)  *
' *                                                                      *
' *  GIVEN:     pmxDoc     = the active view                             *
' *              theTheme  = theme to be processed                       *
' *               aField1    = first field name that theme is to be      *
' *                            classified upon                           *
' *               aField2    = second field name that theme is to be     *
' *                              classified upon, specify NULL if no field *
' *                             is to be used in the classification      *
' *               aField3    = third field name that theme is to be      *
' *                              classified upon, specify NULL if no field *
' *                             is to be used in the classification      *
' *                showNulls = flag denoting whether features that have  *
' *                             not been assigned a value for the        *
' *                              specified fields should be drawn or not *
' *                           (true, false)                              *
' *                                                                      *
' *  RETURN:    nothing                                                  *
' *                                                                      *
' *   NOTE:       (a) All features in the theme will be classified such  *
' *                    that features having a unique value, within the   *
' *                    fields, will be drawn in a unique or different    *
' *                    symbol from the other unique values within the    *
' *                  fields                                              *
' *               (b) After a theme has been classified you must use     *
' *                    avGetLegend to get the new legend that reflects   *
' *                    the new classification if you wish to manipulate  *
' *                    the labels or symbols in the classification       *
' *                                                                      *
' *  Dim pmxDoc As IMxDocument                                           *
' *  Dim theTheme As Variant                                             *
' *  Dim aField1, aField2, aField3 As String                             *
' *  Dim showNulls As Boolean                                            *
' *                                                                      *
Public  Function  avUnJoinAll(aVTab)  As  Integer
' *                                                                      *
' *  PURPOSE:  TO REMOVE ALL JOINS FROM A VTAB                           *
' *                                                                      *
' *  GIVEN:     aVTab        = name of VTab to be processed              *
' *                                                                      *
' *  RETURN:    avUnjoinAll = error flag                                 *
' *                             0 : no error                             *
' *                             1 : error detected                       *
' *                             2 : VTab has no joins                    *
' *                                                                      *
' *  Dim aVTab As String                                                 *
' *  Dim avUnJoinAll As Integer                                          *
' *                                                                      *
Public  Function  avUnLinkAll(aVTab)  As  Integer
' *                                                                      *
' *  PURPOSE:  TO REMOVE ALL LINKS (RELATES) FROM A VTAB                 *
' *                                                                      *
' *  GIVEN:     aVTab        = name of VTab to be processed              *
' *                                                                      *
' *  RETURN:    avUnLinkAll = error flag                                 *
' *                             0 : no error                             *
' *                             1 : error detected                       *
' *                                                                      *
' *  Dim aVTab As String                                                 *
' *  Dim avUnLinkAll As Integer                                          *
' *                                                                      *
```

```
Public  Sub  avUpdateAnno(pFeature  As  IFeature,  _
                          oldX, oldY, newX, newY, _
                          rotang, scaleX, scaleY, _
                          newFeature As IFeature)
' *                                                                        *
' *   PURPOSE:   TO TRANSFORM AN EXISTING ANNOTATION FEATURE               *
' *                                                                        *
' *   GIVEN:     pFeature    = annotation feature to be processed          *
' *              oldX        = x coordinate of feature control point       *
' *              oldY        = y coordinate of feature control point       *
' *              newX        = x coordinate control point to be moved to   *
' *              newY        = y coordinate control point to be moved to   *
' *              rotang      = rotation angle to be applied (degrees)      *
' *                            0.0 : do not rotate feature                 *
' *                            <>0 : add rotation angle to feature         *
' *              scaleX      = X scale factor (can not be <= 0.0)          *
' *              scaleY      = Y scale factor (can not be <= 0.0)          *
' *                                                                        *
' *   RETURN:    newFeature = feature after transformation applied         *
' *                                                                        *
' *   NOTE:      (a) The rotation angle is added to the existing angle     *
' *                  of the annotation (positive value denotes counter-    *
' *                  clockwise rotation, negative value clockwise)         *
' *              (b) Scale factor greater than 1.0 increases the size,     *
' *                  while a value less than 1.0 decreases the size        *
' *              (c) The X scale factor is always used in the scaling      *
' *                  process, the Scale method does not seem to work as    *
' *                  it should on Annotation features when the X and Y     *
' *                  scale factors are different                           *
' *              (d) The layer that the feature resides must be in an      *
' *                  editable state                                        *
' *                                                                        *
' *  Dim pFeature As IFeature                                              *
' *  Dim oldX, oldY, newX, newY, rotang, scaleX, scaleY As Double          *
' *  Dim newFeature As IFeature                                            *
' *                                                                        *
Public  Sub  avUpdateDefaultFont()
' *                                                                        *
' *   PURPOSE:   UPDATE THE DEFAULT FONT PARAMETERS IN THE DRAW TOOLBAR    *
' *              BY EXECUTING THE SELECT ELEMENTS TOOL                     *
' *                                                                        *
' *   GIVEN:     nothing                                                   *
' *                                                                        *
' *   RETURN:    nothing                                                   *
' *                                                                        *
Public  Function  avUpdateJoin(aVTab1,  aVTab2)  As  Integer
' *                                                                        *
' *   PURPOSE:   TO UPDATE THE SELECTION SET IN aVTab2 TO REFLECT THE      *
' *              SELECTION SET OF aVTab1 BASED UPON A JOIN (RELATE)        *
' *                                                                        *
' *   GIVEN:     aVTab1       = name of VTab which aVTab2 is joined to     *
' *              aVTab2       = name of VTab joined to aVTab1              *
' *                                                                        *
' *   RETURN:    avUpdateJoin = error flag                                 *
' *                             0 : no error                               *
' *                             1 : error detected                         *
' *                             2 : aVTab1 does not exist                  *
' *                             3 : aVTab2 does not exist                  *
' *                             4 : join was not found                     *
' *                                                                        *
' *   NOTE:      This procedure will refresh the selection set for the     *
' *              VTab being processed, aVTab1 in addition to the           *
' *              selection set of aVTab2                                   *
' *                                                                        *
' *  Dim aVTab1, aVTab2 As String                                          *
' *  Dim avUpdateJoin As Integer                                           *
' *                                                                        *
```

Public Sub avUpdateLegend(pmxDoc As IMxDocument, theTheme)

```
' *                                                                     *
' *  PURPOSE:  TO UPDATE A THEME TO REFLECT ANY CHANGES MADE TO ITS     *
' *              LEGEND, BOTH THE THEME AND THE TABLE OF CONTENTS WILL  *
' *             BE UPDATED (REDRAWN)                                    *
' *                                                                     *
' *  GIVEN:     pmxDoc   = the active view                              *
' *             theTheme = theme to be processed                        *
' *                                                                     *
' *  RETURN:    nothing                                                 *
' *                                                                     *
' *  NOTE:       It may be necessary to follow the call to the routine  *
' *              avUpdateLegend with a call to avDisplayInvalidate or   *
' *              avGetDisplayFlush in order to refresh the display so   *
' *             that the changes made to the theme are properly         *
' *             displayed, these calls are not made here thereby        *
' *             eliminating multiple screen redraws                     *
' *                                                                     *
' *  Dim pmxDoc As IMxDocument                                          *
' *  Dim theTheme As Variant                                            *
' *                                                                     *
```

Public Function avUpdateLink(aVTab1, aVTab2, aLink) As Integer

```
' *                                                                     *
' *  PURPOSE:  TO UPDATE THE SELECTION SET IN aVTab2 TO REFLECT THE     *
' *              SELECTION SET OF aVTab1 BASED UPON A SPECIFIED LINK    *
' *             (RELATE)                                                *
' *                                                                     *
' *  GIVEN:     aVTab1        = name of VTab which aVTab2 is linked to  *
' *             aVTab2        = name of VTab linked to aVTab1           *
' *             aLinkI        = link number in aVTab1 to be updated     *
' *                               (if aVTab2 is a layer and if aLinkI is *
' *                               negative the selection set of aVTab2 is *
' *                               updated but the display of the selected *
' *                             features is not)                        *
' *                                                                     *
' *  RETURN:    avUpdateLink = error flag                               *
' *                             0 : no error                            *
' *                             1 : error detected                      *
' *                             2 : aVTab1 does not exist               *
' *                             3 : aVTab2 does not exist               *
' *                             4 : link number was not found           *
' *                                                                     *
' *  Dim aVTab1, aVTab2 As String                                       *
' *  Dim aLinkI As Long                                                 *
' *  Dim avUpdateLink As Integer                                        *
' *                                                                     *
```

Public Function avUpdateLinks(aVTab1) As Integer

```
' *                                                                     *
' *  PURPOSE:  TO UPDATE THE SELECTION SETS IN ALL VTabs THAT ARE       *
' *             LINKED (RELATED) TO aVTab1                              *
' *                                                                     *
' *  GIVEN:     aVTab1         = name of VTab to be processed           *
' *                                                                     *
' *  RETURN:    avUpdateLinks = error flag                              *
' *                              0 : no error                           *
' *                              1 : error detected                     *
' *                              2 : aVTab1 does not exist              *
' *                              3 : no links were found                *
' *                                                                     *
' *  NOTE:       This procedure will refresh the selection set for the  *
' *              VTab being processed, aVTab1 in addition to all of the *
' *              selection sets that aVTab1 has links (relates) with    *
' *                                                                     *
' *  Dim aVTab1 As String                                               *
' *  Dim avUpdateLinks As Integer                                       *
' *                                                                     *
```

```
Public Sub avUpdateSelection(pmxDoc As IMxDocument, theTheme)
' *                                                                    *
' *   PURPOSE:   TO UPDATE THE ATTRIBUTE TABLE FOR A THEME TO REFLECT  *
' *              THE CURRENT SELECTION SET FOR THE THEME               *
' *                                                                    *
' *  GIVEN:     pmxDoc    = the active view                            *
' *             theTheme = theme to be processed                       *
' *                                                                    *
' *  RETURN:    nothing                                                *
' *                                                                    *
' *  Dim pmxDoc As IMxDocument                                         *
' *  Dim theTheme As Variant                                           *
' *                                                                    *
Public Sub avViewAddGraphic(pElement As IElement)
' *                                                                    *
' *   PURPOSE:   TO ADD A GRAPHIC INTO THE ACTIVE GRAPHICS LAYER       *
' *                                                                    *
' *  GIVEN:     pElement = graphic to be added                         *
' *                                                                    *
' *  RETURN:    nothing                                                *
' *                                                                    *
' *   NOTE:      Use the subroutine avSetGraphicsLayer to set the      *
' *               active graphics layer (annotation target layer)      *
' *                                                                    *
' *  Dim pElement As IElement                                          *
' *                                                                    *
Public Sub avViewGetGraphics(graList)
' *                                                                    *
' *   PURPOSE:   TO GET A LIST OF ALL OF THE GRAPHICS IN THE MAP       *
' *                                                                    *
' *  GIVEN:     nothing                                                *
' *                                                                    *
' *   RETURN:    graList = list of all graphic elements in the map     *
' *                                                                    *
' *  Dim graList As New Collection                                     *
' *                                                                    *
Public Function avViewMake() As IMap
' *                                                                    *
' *   PURPOSE:   TO CREATE A NEW VIEW (DATA FRAME)                     *
' *                                                                    *
' *  GIVEN:     nothing                                                *
' *                                                                    *
' *   RETURN:    avViewMake = the new view (data frame) object         *
' *                                                                    *
' *   NOTE:      If the view can not be created, avViewMake will be    *
' *             set to NOTHING                                         *
' *                                                                    *
' *  Dim avViewMake As IMap                                            *
' *                                                                    *
Public Function avVTabExport(aTable, aFileName, aClass, selRecrds) _
                                                       As ITable
' *                                                                    *
' *   PURPOSE:   EXPORT AN EXISTING TABLE TO CREATE A NEW TABLE THAT IS *
' *             OF dBASE OR TEXT FILE TYPE                             *
' *                                                                    *
' *  GIVEN:     aTable        = name of the table to be exported       *
' *              aFileName     = name of the table to be created,      *
' *                               if the name does not contain a       *
' *                                complete pathname the current working *
' *                                directory will be used, some examples *
' *                              of name include:                      *
' *                                  c:\project\test\atable            *
' *                                   c:\project\test\atable.dbf       *
' *                                atable                              *
' *                                atable.dbf                          *
' *                               the name can or can not contain the  *
' *                              extension .dbf or .txt                *
' *             aClass        = type of table to be created            *
' *                             dBase                                  *
' *                             TEXT                                  *
' *              selRecrds     = indicates if the selected records are *
' *                             to be exported                        *
' *                               true  = export selected records only *
' *                              false = export all records           *
' *                                                                   *
' *   RETURN:    avVTabExport = table object that is created          *
' *                                                                   *
' *   NOTE:      (a) if the table to be created, aFileName, exists on *
' *                   disk, it will be deleted before the exporting is *
' *                   performed without informing the user/developer  *
' *               (b) if selected records are to be exported and there *
' *                   are no selected records, the entire table will be *
' *                 exported                                          *
' *              (c) if the table can not be exported for any reason  *
' *                   what so ever, avVTabExport will be set to NOTHING *
' *                                                                   *
' *   Dim aTable, aFileName, aClass As String                         *
' *   Dim selRecrds As Boolean                                        *
' *   Dim avVTabExport As ITable                                      *
' *                                                                   *
Public  Function  avVTabMake(aFileName,  forWrite,  skipFirst,  _
                              Optional aClass) As ITable
' *                                                                   *
' *   PURPOSE:   OPEN AN EXISTING TABLE THAT IS OF dBASE OR TEXT FILE  *
' *             TYPE                                                  *
' *                                                                   *
' *   GIVEN:     aFileName      = name of the table to be opened,      *
' *                               if the name does not contain a      *
' *                                complete pathname the current working *
' *                                directory will be used, some examples *
' *                              of name include:                     *
' *                                   c:\project\test\atable.dbf      *
' *                                 atable.dbf                        *
' *                                the extension .dbf or .txt indicates *
' *                               the type of table to be opened      *
' *              forWrite       = indicates if the table is to be made *
' *                               editable once it is opened          *
' *              skipFirst      = indicates if the first record in the *
' *                              table is to be ignored               *
' *              aClass          = optional argument which specifies the *
' *                               type of table to be opened          *
' *                             dBase                                 *
' *                             TEXT                                  *
' *                                if this argument is specified it will *
' *                                override any extension that may appear *
' *                              in aFileName                         *
' *                                                                   *
' *   RETURN:    avVTabMake     = table object that is created        *
' *                                                                   *
' *   NOTE:      (a) The forWrite and skipFirst arguments are ignored, *
' *                   as of this version, and as such have no impact  *
' *                 upon this procedure                               *
' *               (b) If aFileName does not contain an extension the  *
' *                   procedure assumes a dBase file is to be opened  *
' *               (c) If aFileName can not be opened, avVTabMake will be *
' *                 set to NOTHING                                    *
' *               (d) Use the function avAddDoc to add the table into *
' *                  the Table of Contents                            *
' *                                                                   *
' *   Dim aFileName As String                                         *
' *   Dim forWrite, skipFirst As Boolean                              *
' *   Dim aClass As String                                            *
' *   Dim avVTabMake As ITable                                        *
' *                                                                   *
Public  Function  avVTabMakeNew(aFileName,  aclass)  As  ITable
' *                                                                   *
' *   PURPOSE:   CREATE A NEW TABLE THAT IS OF dBASE OR TEXT FILE TYPE *
' *                                                                   *
' *  GIVEN:     aFileName      = name of the table to be created,       *
' *                              if the name does not contain a         *
' *                               complete pathname the current working *
' *                               directory will be used, some examples *
' *                             of name include:                        *
' *                                 c:\project\test\atable              *
' *                                  c:\project\test\atable.dbf         *
' *                                atable                               *
' *                                atable.dbf                           *
' *                              the name can or can not contain the    *
' *                             extension .dbf or .txt                  *
' *             aClass         = type of table to be created            *
' *                             dBase                                   *
' *                             TEXT                                    *
' *                                                                     *
' *  RETURN:    avVTabMakeNew = table object that is created            *
' *                                                                     *
' *  NOTE:       (a) Two fields called OID and ID will be created by    *
' *                   this routine, the function avAddDoc can be used to *
' *                  add the table to the map, if need be               *
' *               (b) If the table to be created exists on disk, the    *
' *                   routine will abort the existing table will not be *
' *                 overwritten                                         *
' *                                                                     *
' *  Dim aFileName, aClass As String                                    *
' *  Dim avVTabMakeNew As ITable                                        *
' *                                                                     *
```
Public Sub avXORSelection(pmxDoc As IMxDocument, theTheme, _
orgSelSet As ISelectionSet, _
xorSelSet As ISelectionSet)
```
' *                                                                     *
' *  PURPOSE:   PERFORM AN EXCLUSIVE XOR ON A LAYER OR TABLE SELECTION  *
' *             SET                                                     *
' *                                                                     *
' *  GIVEN:     pmxDoc     = the active view                            *
' *               theTheme   = the theme or table to be processed       *
' *              orgSelSet = the selection set to be XOR                *
' *                                                                     *
' *  RETURN:     xorSelSet = the selection set after XOR was performed  *
' *                                                                     *
' *  NOTE:       The current selection set for the theme or table is    *
' *               used in the XORing with the set that is passed in     *
' *               (orgSelSet). If theTheme has no selection set, the    *
' *               command will select all features (rows) in theTheme   *
' *              and use this set in the XOR process                    *
' *                                                                     *
' *  Dim pmxDoc As IMxDocument                                          *
' *  Dim theTheme As Variant                                            *
' *  Dim orgSelSet As ISelectionSet                                     *
' *  Dim xorSelSet As ISelectionSet                                     *
' *                                                                     *
```
Public Sub avZoomToSelected(pmxDoc As IMxDocument, theTheme)
```
' *                                                                     *
' *  PURPOSE:   TO ZOOM TO THE EXTENT OF THE SELECTED SET FOR A THEME   *
' *               OR THE EXTENT OF ALL SELECTED FEATURES IN THE MAP     *
' *                                                                     *
' *  GIVEN:     pmxDoc    = the active view                             *
' *               theTheme = theme for which its selected features will *
' *                          be zoomed to, if NULL is specified all     *
' *                          selected features will be zoomed to        *
' *                                                                     *
' *  RETURN:    nothing                                                 *
' *                                                                     *
' *  NOTE:       If a theme is specified and the theme does not contain *
' *               any selected features, the command will zoom to the   *
' *               full extent of the theme (all features processed in   *
' *             this condition)                                         *
' *                                                                     *
' *  Dim pmxDoc As IMxDocument                                          *
' *  Dim theTheme As Variant                                           *
' *                                                                    *
Public  Sub  avZoomToTheme(pmxDoc  As  IMxDocument,  theTheme)
' *                                                                    *
' *  PURPOSE:  TO ZOOM TO THE EXTENT OF A THEME                        *
' *                                                                    *
' *  GIVEN:     pmxDoc   = the active view                             *
' *             theTheme = theme to be processed                       *
' *                                                                    *
' *  RETURN:   nothing                                                 *
' *                                                                    *
' *  Dim pmxDoc As IMxDocument                                         *
' *  Dim theTheme As Variant                                           *
' *                                                                    *
Public  Sub  avZoomToThemes(pmxDoc  As  IMxDocument,  thmList)
' *                                                                    *
' *  PURPOSE:  TO ZOOM TO THE EXTENT OF A GROUP OF THEMES              *
' *                                                                    *
' *  GIVEN:     pmxDoc  = the active view                              *
' *             thmList = list of themes to be processed               *
' *                                                                    *
' *  RETURN:   nothing                                                 *
' *                                                                    *
' *  Dim pmxDoc As IMxDocument                                         *
' *  Dim thmList As New Collection                                     *
' *                                                                    *
Public  Sub  ChangeView(pmxDoc  As  IMxDocument,  opmode,  _
                           sclFctr, panXval, panYval, usrView As IUnknown, _
                          iok, newRect As IEnvelope)
' *                                                                    *
' *  PURPOSE:  SCRIPT TO ALTER THE DISPLAY OF THE VIEW                 *
' *                                                                    *
' *  GIVEN:     pmxDoc  = the active view                              *
' *             opmode  = mode of operation                            *
' *                         1 : zoom scale factor to be applied        *
' *                         2 : panning values to be applied           *
' *                         3 : a new extent to be defined             *
' *                         4 : center display about a point           *
' *             sclFctr = scale factor to be applied to view           *
' *             panXval = distance in world units to pan along x axis  *
' *                       or display center x coordinate if opmode = 4 *
' *             panYval = distance in world units to pan along y axis  *
' *                       or display center y coordinate if opmode = 4 *
' *             usrView = user-defined view extent rectangle, can be   *
' *                       either an IPolygon or IEnvelope object       *
' *                                                                    *
' *  RETURN:    iok     = error flag (0 = no error, 1 = error)         *
' *             newRect = view extent rectangle                        *
' *                                                                    *
' *  Dim pmxDoc As IMxDocument                                         *
' *  Dim opmode As Integer                                             *
' *  Dim sclFctr, panXval, panYval As Double                           *
' *  Dim usrView As IUnknown                                           *
' *  Dim iok As Integer                                                *
' *  Dim newRect As IEnvelope                                          *
' *                                                                    *
Public  Sub  CopyList(origList,  newList)
' *                                                                    *
' *  PURPOSE:  TO COPY A COLLECTION INTO ANOTHER COLLECTION AND THEN   *
' *            INITIALIZE OR CLEAR THE ORIGINAL COLLECTION             *
' *                                                                    *
' *  GIVEN:     origList = list to be copied and then cleared          *
' *                                                                    *
' *  RETURN:    newList  = copy of the original list                   *
' *                                                                    *
' *  Dim origList As New Collection                                    *
' *  Dim newList As New Collection                                     *
' *                                                                    *
```

```
Public  Sub  CopyList2(origList,  newList)
' *                                                                           *
' *   PURPOSE:   TO COPY A COLLECTION INTO ANOTHER COLLECTION AND THEN        *
' *               INITIALIZE OR CLEAR THE ORIGINAL COLLECTION                 *
' *                                                                           *
' *   GIVEN:     origList = list to be copied and then cleared                *
' *                                                                           *
' *   RETURN:    newList  = copy of the original list                         *
' *                                                                           *
' *   NOTE:       These are collections of objects, not variables such        *
' *              as doubles, longs, integers, etc.                            *
' *                                                                           *
' *   Dim origList As New Collection                                          *
' *   Dim newList As New Collection                                           *
' *                                                                           *
Public  Sub  CopyList3(origList,  newList)
' *                                                                           *
' *   PURPOSE:   TO COPY A COLLECTION INTO ANOTHER COLLECTION LEAVING         *
' *               THE ORIGINAL COLLECTION INTACT (UNALTERED)                  *
' *                                                                           *
' *   GIVEN:     origList = list to be copied                                 *
' *                                                                           *
' *   RETURN:    newList  = copy of the original list                         *
' *                                                                           *
' *   Dim origList As New Collection                                          *
' *   Dim newList As New Collection                                           *
' *                                                                           *
Public  Function  CreateAccessDB(sDir,  sDBName,  bOverWrite)  As  IWorkspace
' *                                                                           *
' *   PURPOSE:  CREATE A PERSONAL GEODATABASE                                 *
' *                                                                           *
' *   GIVEN:     sDir             = directory location                        *
' *               sDBName          = geodatabase name (do not include the     *
' *                                  .mdb extension in the name)              *
' *               bOverWrite      = flag denoting whether to overwrite        *
' *                                 the geodatabase if it exists              *
' *                                 true = overwrite, false = do not          *
' *                                                                           *
' *   RETURN:    CreateAccessDB = geodatabase that is created, will           *
' *                                 be set to NOTHING if an error was         *
' *                                  encountered during the processing        *
' *                                                                           *
' *   NOTE:       (a) The geodatabase created will contain no dataset or      *
' *                    feature class. These will have to be added later       *
' *                   on if need be (only the .mdb file is created)           *
' *               (b) The Editor should not be active, if a personal          *
' *                    geodatabase is being editted and this function is      *
' *                    called during the editting, an automation error        *
' *                    will be generated. This is why avStopEditing is        *
' *                    called, to make sure the Editor is not in an edit      *
' *                  state                                                    *
' *                                                                           *
' *   Dim sDir As String                                                      *
' *   Dim sDBName As String                                                   *
' *   Dim bOverWrite As Boolean                                               *
' *   Dim CreateAccessDB As IWorkspace                                        *
' *                                                                           *
Public  Function  CreateAnnoClass(pWorkspace  As  IWorkspace,  _
                              theName, pFields As IFields, _
                              dRefScale, dUnits) As IFeatureClass
' *                                                                           *
' *   GIVEN:     pWorkspace       = connection to the geodatabase             *
' *               theName          = annotation feature class name if the     *
' *                                   feature class is to appear in a         *
' *                                    dataset of the same name, otherwise,   *
' *                                    the annotation feature class name      *
' *                                    and the name of the dataset in which   *
' *                                    the feature class is to appear in      *
' *              pFields          = shapefile attributes                      *
' *             dRefScale        = reference scale of the current view  *
' *             dUnits           = units setting of the current view    *
' *                                                                     *
' *  RETURN:    CreateAnnoClass  = feature class that is created, will  *
' *                                be set to NOTHING if an error was    *
' *                                encountered during the processing    *
' *                                                                     *
' *  NOTE:      (a) Do not use the hyphen or dash (-) character in      *
' *                 the name for the annotation feature class           *
' *             (b) The first character in the feature class name       *
' *                 should not be numeric (not a number, but rather     *
' *               an alphacharacter)                                    *
' *             (c) When theName contains both the feature class and    *
' *                 dataset names at least one space must separate the *
' *                 two items (the feature class precedes the dataset) *
' *             (d) If the dataset in which the feature class is to     *
' *                 appear in does not exist, it will be created, if    *
' *                 the dataset does exist, it will be used as is       *
' *                                                                     *
' *  Dim pWorkspace As IWorkspace                                       *
' *  Dim theName As String                                              *
' *  Dim pFields As IFields                                             *
' *  Dim dRefScale As Double                                            *
' *  Dim dUnits As esriUnits                                            *
' *  Dim CreateAnnoClass As IFeatureClass                               *
' *                                                                     *
Public  Function  CreateFeatClass(pFeatureDataset  _
                                  As IFeatureDataset, theName, _
                                  geomType As esriGeometryType, _
                                  Optional pFields As IFields) _
                                  As IFeatureClass
' *                                                                     *
' *  PURPOSE:   CREATE A FEATURE CLASS WITHIN A FEATURE DATASET IN A    *
' *             GEODATABASE                                             *
' *                                                                     *
' *  GIVEN:     pFeatureDataset = feature dataset to be processed       *
' *             theName         = featureclass name                     *
' *             geomType        = featureclass geometry type, such as: *
' *                               esriGeometryPoint                     *
' *                               esriGeometryPolyline                  *
' *                               esriGeometryPolygon                   *
' *             pFields         = feature attributes (optional)         *
' *                                                                     *
' *  RETURN:    CreateFeatClass = feature class that is created, will  *
' *                               be set to NOTHING if an error was     *
' *                               encountered during the processing     *
' *                                                                     *
' *  NOTE:      (a) The function CreateNewShapefile can be used to      *
' *                 create the IFeatureDataset object, if appropriate  *
' *             (b) If pFields contains any geometry fields they will   *
' *                 be ignored, only valid attribute fields will be     *
' *               processed                                             *
' *             (c) If pFields contains the Shape field, the settings   *
' *                 for the HasM and HasZ properties will be used in    *
' *               creating the feature class                            *
' *             (d) If pFields is not specified only the OID and SHAPE *
' *                 fields will be added to the featureclass            *
' *             (e) Do not use the hyphen or dash (-) character in      *
' *                 the name for the annotation feature class           *
' *             (f) The first character in the feature class name       *
' *                 should not be numeric (not a number, but rather     *
' *               an alphacharacter)                                    *
' *                                                                     *
' *  Dim pFeatureDataset As IFeatureDataset                             *
' *  Dim theName As String                                              *
' *  Dim geomType As esriGeometryType                                   *
' *  Dim pFields As IFields                                             *
' *  Dim CreateFeatClass As IFeatureClass                               *
' *                                                                     *
```

```
Public  Sub  CreateList(newList)
' *                                                                     *
' *   PURPOSE:   TO CREATE A NEW COLLECTION MAKING SURE IT IS EMPTY      *
' *                                                                     *
' *  GIVEN:     nothing                                                 *
' *                                                                     *
' *  RETURN:    newList = new list                                      *
' *                                                                     *
' *  Dim newList As New Collection                                      *
' *                                                                     *
Public  Function  CreateNewGeoDB(pFieldsI  As  IFields,  _
                                 geomType As esriGeometryType, _
                                 defNameI As String, _
                                 aTitle As String) As IFeatureClass
' *                                                                     *
' *   PURPOSE:   USING A FILE DIALOG BOX PROMPT THE USER TO SPECIFY A    *
' *               WORKSPACE, WHICH WILL BE A PERSONAL GEODATABASE.      *
' *                ALTERNATIVELY, THE PERSONAL GEODATABASE CAN BE CREATED *
' *              WITHOUT ANY USER INTERACTION                           *
' *                                                                     *
' *  GIVEN:     pFieldsI        = attributes to be stored in the        *
' *                                 new personal geodatabase            *
' *              geomType        = shapefile geometry type              *
' *                                  (as of this implementation not used, *
' *                                 so that, specify as NOTHING)        *
' *               defNameI        = default filename (see notes below)  *
' *              aTitle          = file dialog message box title, if    *
' *                                 aTitle is equal to CREATEandLOAD    *
' *                                 no file dialog box will be shown,   *
' *                                 the shapefile or PGD will be        *
' *                                 created without user intervention   *
' *                                                                     *
' *   RETURN:    CreateNewGeoDB = feature class that is created, will   *
' *                                 be set to NOTHING if an error was   *
' *                                 encountered during the processing   *
' *                                                                     *
' *   NOTE:       File Dialog is Displayed (aTitle <> "CREATEandLOAD")  *
' *               (a) A stand-alone annotation feature class within a   *
' *                    feature dataset is created by this function, the *
' *                    names of the feature dataset and the annotation  *
' *                   feature class are the same (see note b)           *
' *              (b) Optionally, the user can enter up to 3 names in    *
' *                   the file name data entry field, separated by at   *
' *                    least one space (blank character) when a personal *
' *                    geodatabase is to be created. When 1 name is given *
' *                    see note c. When 2 names are specified, the first *
' *                    name defines the name of the dataset and feature *
' *                    class while the second defines the name of the PGD *
' *                   to be created. When 3 names are specified, the    *
' *                    first defines the name of the feature class, the *
' *                   second defines the name of the dataset and the    *
' *                   third defines the name of the PGD to be created   *
' *               (c) Use CreateNewShapefile specifying the .mdb file   *
' *                    name extension in the default filename to create *
' *                    a geodatabase that contains a feature class and  *
' *                  not an annotation feature class                    *
' *               (d) The new annotation feature class is automatically *
' *                   added to the map once it has been created         *
' *               (e) If an existing .mdb file is selected, the user can *
' *                    either abort the command (CANCEL), add to the .mdb *
' *                    file (NO) or overwrite the existing file (YES)   *
' *              (f) When an existing .mdb file is appended the root    *
' *                   name of the default filename is used as the name  *
' *                   of the new annotation class that is created       *
' *               (g) When an existing .mdb file is to be overwritten,  *
' *                   if the file exists in the map the function will   *
' *                   not delete the file but will inform the user and  *
' *                 abort the function                                  *
' *              (h) The Map Units for the data frame must be set to    *
' *                     something other than Unknown Units, otherwise the   *
' *                     MapScale property will result in an automation      *
' *                   error message being generated                         *
' *               File Dialog not Displayed (aTitle = "CREATEandLOAD")      *
' *               (i) When aTitle = "CREATEandLOAD" this denotes that       *
' *                     the default filename (defNameI) is to be created    *
' *                     and loaded without displaying the file dialog box   *
' *                     In this mode of operation, defNameI can contain     *
' *                   up to three items separated by a space:               *
' *                   >>>Single Item condition<<<                           *
' *                     Under this condition, the programmer specifies      *
' *                     the name of the personal geodatabase to be created *
' *                    and loaded. A full pathname for the personal         *
' *                    geodatabase must be given. If the personal           *
' *                     geodatabase exists, it will not be deleted but      *
' *                     rather, it will be used as is. The programmer has   *
' *                     to make sure that the personal geodatabase does     *
' *                     not already exist in the map, otherwise, multiple   *
' *                     copies of the personal geodatabase will appear in   *
' *                     the TOC because the existing  personal geodatabase *
' *                   will be loaded into the map                           *
' *                     Example of defNameI to create a geodatabase that    *
' *                     will be named L_0.mdb and will contain a feature    *
' *                     dataset and an annotation feature class named L_0   *
' *                       defNameI = "c:\temp\L_0.mdb"                      *
' *                   >>>Two Item condition<<<                              *
' *                     Under this condition, the programmer specifies      *
' *                     the name of a feature dataset to be created and a   *
' *                     personal geodatabase in which the feature dataset   *
' *                     is to be stored in. The personal geodatabase can    *
' *                    either exist or not, if it does not it will be       *
' *                     created. If the personal geodatabase exists, the    *
' *                    feature dataset will be added to the personal        *
' *                     geodatabase. If the feature dataset exists in the   *
' *                     personal geodatabase, it will not be deleted but    *
' *                     rather, it will be used as is. The programmer has   *
' *                     to make sure the feature dataset does not already   *
' *                     exist in the map, otherwise, multiple copies of     *
' *                     the feature dataset will appear in the TOC because *
' *                     the existing feature dataset will be loaded into    *
' *                  the map                                                *
' *                     Example of defNameI to create a geodatabase that    *
' *                     will be named L_0.mdb and will contain a feature    *
' *                     dataset and annotation feature class named G_Grid   *
' *                        defNameI = "G_Grid c:\temp\L_0.mdb"              *
' *                   >>>Three Item condition<<<                            *
' *                     Similar to the two item condition described above   *
' *                     with the exception that the user can control the    *
' *                   name of the dataset that is created.                  *
' *                     Example of defNameI to create a geodatabase that    *
' *                     will be named L_0.mdb and will contain a feature    *
' *                     dataset called Profile and an annotation feature    *
' *                  class named G_Grid                                     *
' *                         defNameI = "G_Grid Profile c:\temp\L_0.mdb"     *
' *                                                                         *
' *  Dim pFieldsI As IFields                                                *
' *  Dim geomType As esriGeometryType                                       *
' *  Dim defNameI As String                                                 *
' *  Dim aTitle As String                                                   *
' *  Dim CreateNewGeoDB As IFeatureClass                                    *
' *                                                                         *
Public  Function  CreateNewShapefile(pFieldsI  As  IFields,  _
                        geomType As esriGeometryType, defNameI As String, _
                        aTitle As String) As IFeatureClass
' *                                                                         *
' *   PURPOSE:   USING A FILE DIALOG BOX PROMPT THE USER TO SPECIFY A       *
' *               WORKSPACE TO BE CREATED, THIS CAN BE A SHAPEFILE OR A     *
' *                PERSONAL GEODATABASE. ALTERNATIVELY, THE SHAPEFILE OR    *
' *               PERSONAL GEODATABASE CAN BE ESTABLISHED WITHOUT ANY       *
' *              USER INTERACTION                                       *
' *                                                                     *
' *  GIVEN:     pFieldsI             = attributes to be stored in the   *
' *                                    new shapefile                    *
' *              geomType             = shapefile geometry type, such as: *
' *                                    esriGeometryPoint                *
' *                                    esriGeometryPolyline             *
' *                                    esriGeometryPolygon              *
' *                                     (use CreateNewGeoDB when dealing *
' *                                     with annotation features)       *
' *              defNameI             = default filename, this may or may *
' *                                     not contain a filename extension *
' *                                    (see note a below)               *
' *              aTitle               = file dialog message box title, if *
' *                                     aTitle is equal to CREATEandLOAD *
' *                                     no file dialog box will be shown, *
' *                                     the shapefile or PGD will be    *
' *                                     created without user intervention *
' *                                                                     *
' *  RETURN:    CreateNewShapefile =  feature class that is created,    *
' *                                    will be set to NOTHING if an     *
' *                                     error was encountered during the *
' *                                    processing                       *
' *                                                                     *
' *  NOTE:       File Dialog is Displayed (aTitle <> "CREATEandLOAD")   *
' *               (a) If the defName argument contains the .shp filename *
' *                    extension, the dataset type that will be created *
' *                    will be a shapefile. If the .mdb filename extension*
' *                    is found, the type of dataset created will be a  *
' *                    personal geodatabase. If no filename extension is *
' *                    given both types will appear in the list of      *
' *                    available types and the user can pick the desired *
' *                  type.                                              *
' *               (b) The new shapefile or geodatabase is automatically *
' *                    added to the map once it has been created        *
' *               (c) When a personal geodatabase is created a feature  *
' *                    dataset and a feature class are created using the *
' *                    same name, the feature class is added to the     *
' *                   feature dataset (see note d)                      *
' *               (d) Optionally, the user can enter up to 3 names in   *
' *                    the file name data entry field, separated by at  *
' *                    least one space (blank character) when a personal *
' *                    geodatabase is to be created. When 1 name is given *
' *                    see note c. When 2 names are specified, the first *
' *                    name defines the name of the dataset and feature *
' *                    class while the second defines the name of the PGD *
' *                    to be created. When 3 names are specified, the   *
' *                    first defines the name of the feature class, the *
' *                    second defines the name of the dataset and the   *
' *                    third defines the name of the PGD to be created  *
' *               (e) If an existing .mdb file is selected, the user can *
' *                    either abort the command (CANCEL), add to the .mdb *
' *                    file (NO) or overwrite the existing file (YES)   *
' *               (f) When an existing .mdb file is appended the root   *
' *                    name of the default filename is used as the name *
' *                    of the new feature class that is created         *
' *               (g) When an existing .mdb file is to be overwritten,  *
' *                    if the file exists in the map the function will  *
' *                    not delete the file but will inform the user and *
' *                  abort the function                                 *
' *               File Dialog not Displayed (aTitle = "CREATEandLOAD")  *
' *               (h) When aTitle = "CREATEandLOAD" this denotes that   *
' *                    the default filename (defNameI) is to be created *
' *                    and loaded without displaying the file dialog box *
' *                    In this mode of operation, defNameI can contain  *
' *                   up to three items separated by a space:           *
' *                   >>>Single Item condition<<<                       *
' *                    Under this condition, the programmer specifies   *
' *                    the name of the shapefile or personal geodatabase *
' *                   to be created and loaded. A full pathname for the   *
' *                   shapefile or personal geodatabase must be given     *
' *                   If the shapefile or personal geodatabase exists,    *
' *                   it will not be deleted but rather, it will be used *
' *                   as is. The programmer has to make sure that the     *
' *                   shapefile or personal geodatabase does not already *
' *                   exist in the map, otherwise, multiple copies of     *
' *                   the shapefile or personal geodatabase will appear   *
' *                   in the TOC because the existing shapefile or        *
' *                   personal geodatabase will be loaded into the map    *
' *                   Example of defNameI to create a shapefile that      *
' *                 will be named L_0.shp                                 *
' *                     defNameI = "c:\temp\L_0.shp"                      *
' *                   Example of defNameI to create a geodatabase that    *
' *                   will be named L_0.mdb and will contain a feature    *
' *                  dataset and feature class named L_0                  *
' *                     defNameI = "c:\temp\L_0.mdb"                      *
' *                 >>>Two Item condition<<<                              *
' *                   Under this condition, the programmer specifies      *
' *                   the name of a feature dataset to be created and a   *
' *                   personal geodatabase in which the feature dataset   *
' *                   is to be stored in. The personal geodatabase can    *
' *                   either exist or not, if it does not it will be      *
' *                   created. If the personal geodatabase exists, the    *
' *                   feature dataset will be added to the personal       *
' *                   geodatabase. If the feature dataset exists in the   *
' *                   personal geodatabase, it will not be deleted but    *
' *                   rather, it will be used as is. The programmer has   *
' *                   to make sure the feature dataset does not already   *
' *                   exist in the map, otherwise, multiple copies of     *
' *                   the feature dataset will appear in the TOC because *
' *                   the existing feature dataset will be loaded into    *
' *                 the map                                               *
' *                   Example of defNameI to create a geodatabase that    *
' *                   will be named L_0.mdb and will contain a feature    *
' *                  dataset and feature class named G_Grid               *
' *                     defNameI = "G_Grid c:\temp\L_0.mdb"               *
' *                 >>>Three Item condition<<<                            *
' *                   Similar to the two item condition described above   *
' *                   with the exception that the user can control the    *
' *                  name of the dataset that is created.                 *
' *                   Example of defNameI to create a geodatabase that    *
' *                   will be named L_0.mdb and will contain a feature    *
' *                   dataset called Profile and a feature class named    *
' *                 G_Grid                                                *
' *                      defNameI = "G_Grid Profile c:\temp\L_0.mdb"      *
' *                                                                       *
' *  Dim pFieldsI As IFields                                              *
' *  Dim geomType As esriGeometryType                                     *
' *  Dim defNameI As String                                               *
' *  Dim aTitle As String                                                 *
' *  Dim CreateNewShapefile As IFeatureClass                              *
' *                                                                       *
```

```
Public  Function  CreateShapeFile(featWorkspace  _
                                  As IFeatureWorkspace, _
                                  strName As String, _
                                  geomType As esriGeometryType, _
                                  Optional pFields As IFields, _
                                  Optional pCLSID As UID) As IFeatureClass
```

```
' *                                                                       *
' *   PURPOSE:   CREATE A NEW SHAPEFILE USING INFORMATION EXPLICITLY      *
' *              DEFINED IN THE CALLING ARGUMENTS (NO USER INTERACTION) *
' *                                                                       *
' *  GIVEN:     featWorkspace    = directory location                     *
' *             strName          = shapefile name                         *
' *             geomType         = shapefile geometry type                *
' *                                esriGeometryPoint                      *
' *                                esriGeometryPolyline                   *
' *                                esriGeometryPolygon                    *
' *              pFields           = shapefile attributes (optional)       *
' *              pCLSID            = geometry type subclass (optional)     *
' *                                                                        *
' *   RETURN:    CreateShapeFile = feature class that is created           *
' *                                                                        *
' *   NOTE:       (a) The name of the shapefile should not contain the     *
' *                   .shp extension, if it does it will be stripped off   *
' *                (b) If the pFields argument is not specified a default  *
' *                    shape field with a default spatial reference will   *
' *                   be assigned and one attribute called ID will be      *
' *                  added to the shapefile                                *
' *                                                                        *
' *   Dim featWorkspace As IFeatureWorkspace                               *
' *   Dim strName As String                                                *
' *   Dim geomType As esriGeometryType                                     *
' *   Dim pFields As IFields                                               *
' *   Dim pCLSID As UID                                                    *
' *   Dim CreateShapeFile As IFeatureClass                                 *
' *                                                                        *
Public  Function  Dformat(theNumber,  TotalDigits,  DigitsRight)
' *                                                                        *
' *   PURPOSE:   This is a subroutine type script used to format a data    *
' *               field of an output file according to a Fortran  Fa.b     *
' *              format.                                                   *
' *                The input argument list contains three elements which   *
' *              are expected to be read as numbers                        *
' *                                                                        *
' *   GIVEN:     theNumber    = the real number to be formatted            *
' *                TotalDigits = the number of the data field characters   *
' *                             including leading spaces, decimal point    *
' *                            and decimal digits                          *
' *               DigitsRight = digits to right of decimal point           *
' *                                                                        *
' *   RETURN:    Dformat      = string representing the number in the      *
' *                            specified format                            *
' *                                                                        *
' *   NOTE:       If the number will not fit within the specified data     *
' *               field length as specified by TotalDigits, then the       *
' *               data field will be expanded to accomodate the number.    *
' *                                                                        *
' *   Dim theNumber As Double                                              *
' *   Dim TotalDigits, DigitsRight As Integer                              *
' *   Dim Dformat As String                                                *
' *                                                                        *
Public  Sub  ExportVBAcode()
' *                                                                        *
' *   PURPOSE:   EXPORT VBA COMPONENTS FROM THE CURRENT PROJECT INTO A     *
' *              SPECIFIED DIRECTORY                                       *
' *                                                                        *
' *   GIVEN:     nothing                                                   *
' *                                                                        *
' *   RETURN:    nothing                                                   *
' *                                                                        *
' *   NOTE:       (a) The directory is a hardcoded path stored in aDIR,    *
' *                   if the directory does not exist, it is created       *
' *               (b) The Avenue Wraps: avFileExists and                   *
' *                                     avFileDelete                       *
' *                    must appear in the project file in order for this   *
' *                  macro to execute                                      *
' *                (c) If the component being exported exists on disk, it  *
' *                    will be deleted and the component in the project    *
' *                    file will replace whatever was on disk previously   *
' *                (d) A file called index.txt will be created which will  *
' *                    contain a list of the components that were written  *
' *                  to disk                                               *
' *                                                                        *
Public  Function  FindLayer(map  As  IMap,  name  As  Variant)  As  iLayer
' *                                                                        *
' *   PURPOSE:  FIND A LAYER IN A MAP                                      *
' *                                                                  *
' *  GIVEN:     map        = map to be searched                      *
' *              name       = name of layer to be found               *
' *                                                                  *
' *  RETURN:    FindLayer = the layer in the map                     *
' *                                                                  *
' *  Dim map As IMap                                                 *
' *  Dim name As Variant                                             *
' *  Dim FindLayer As ILayer                                         *
' *                                                                  *
Public  Function  FindTheme(map  As  IMap,  nameIN  As  Variant)  As  Variant
' *                                                                  *
' *  PURPOSE:  FIND A THEME IN A MAP                                 *
' *                                                                  *
' *  GIVEN:     map        = map to be searched                      *
' *              nameIN     = name of theme to be found               *
' *                                                                  *
' *  RETURN:    FindTheme = the theme in the map                     *
' *                                                                  *
' *  Dim map As IMap                                                 *
' *  Dim nameIN As Variant                                           *
' *  Dim FindTheme As Variant                                        *
' *                                                                  *
Public  Sub  GetShape(elmntTheme,  elmntRecrd,  _
                        shapeType, shapeList, shapeDist)
' *                                                                  *
' *   PURPOSE:   TO CREATE A LIST CONTAINING THE COORDINATES OF THE  *
' *              POINTS THAT COMPRISE A FEATURE                      *
' *                                                                  *
' *  GIVEN:     elmntTheme = theme of the feature                    *
' *              elmntRecrd = record number of the feature            *
' *                                                                  *
' *  RETURN:    shapeType  = shape type enumerator                   *
' *               shapeList  = shape's list of points and/or parts    *
' *               shapeDist  = ArcView length of the shape in map units *
' *                                                                  *
' *   NOTE:      This procedure will process a geometry from global  *
' *               memory (ugShapeT or avwraps.ShapeT) when the input  *
' *               arguments, elmntTheme and elmntRecrd are set to be  *
' *               " ", and -1 respectively. Use this approach when the *
' *               geometry for the feature already exists, in so doing *
' *               the developer eliminates a reading of the database  *
' *                                                                  *
' *  Dim elmntTheme As Variant                                       *
' *  Dim elmntRecrd As Long                                          *
' *  Dim shapeType As esriGeometryType                               *
' *  Dim shapeList As New Collection                                 *
' *  Dim shapeDist As Double                                         *
' *                                                                  *
Public  Sub  GetTextFont(pmxDoc  As  IMxDocument,  _
                           fontStrg, currSize, defTFINC, defPMODE, defCOLOR)
' *                                                                  *
' *   PURPOSE:   TO DETERMINE THE CURRENT ACTIVE TEXT FONT IN ARCMAP  *
' *                                                                  *
' *  GIVEN:     pMxDoc    = current active document                  *
' *                                                                  *
' *  RETURN:    fontStrg = name of current active font               *
' *              currSize = font size                                *
' *               defTFINC = font style (1 = normal, 2 = italic)      *
' *               defPMODE = font style (1 = normal, 3 = bold)        *
' *               defCOLOR = font color, RGB color index value        *
' *                                                                  *
' *  Dim pMxDoc As IMxDocument                                       *
' *  Dim fontStrg As String                                          *
' *  Dim currSize As Double                                          *
' *  Dim defTFINC, defPMODE As Integer                               *
' *  Dim defCOLOR As Long                                            *
' *                                                                  *
```

```
Public Sub GetTextRect(pTextElement As ITextElement, _
                        pScreenDisplay As IScreenDisplay, _
                        X1, Y1, aAngle, aWidth, aHeight)
' *                                                                      *
' *  PURPOSE:  TO GET THE ANGLE, HEIGHT AND WIDTH OF A TEXT ELEMENT      *
' *                                                                      *
' *  GIVEN:    pTextElement   = text element representing the text       *
' *            pScreenDisplay = display which text element appears in    *
' *                                                                      *
' *  RETURN:   x1,y1          = low left corner coordinates of text      *
' *            aAngle         = text angle (degrees)                     *
' *            aWidth         = text width (along the text angle)        *
' *            aHeight        = text height (perpendicular to angle)     *
' *                                                                      *
' *  NOTE:     The attributes passed passed back reflect those of an     *
' *            inclined, not an orthogonal, enclosing rectangle that     *
' *            circumscribes the text element                            *
' *                                                                      *
' *  Dim pTextElement As ITextElement                                    *
' *  Dim pScreenDisplay As IScreenDisplay                                *
' *  Dim x1, y1, aAngle, aWidth, aHeight As Double                       *
' *                                                                      *
Public Sub HDBbuild(instruct, Heading, LabelList, DefaultInfo, _
                    TypeList, colmnList, nRows, UserInfo)
' *                                                                      *
' *  PURPOSE:  BUILD A CUSTOMIZABLE HORIZONTAL DIALOG BOX                *
' *                                                                      *
' *  GIVEN:    instruct    = message box instruction                     *
' *            Heading     = message box heading (title)                 *
' *            labelList   = list of column labels to be displayed       *
' *            defaultInfo = list of default values for each column      *
' *            typeList    = list of item data types for each column     *
' *                          1 = data line, 2 = combo box                *
' *            colmnList   = list of column widths                       *
' *            nRows       = number of rows in the dialog box            *
' *                                                                      *
' *  RETURN:   userInfo    = list of user responses for data items       *
' *                                                                      *
' *  Dim instruct, Heading As String                                     *
' *  Dim labelList As New Collection                                     *
' *  Dim defaultInfo As New Collection                                   *
' *  Dim typeList As New Collection                                      *
' *  Dim colmnList As New Collection                                     *
' *  Dim nRows As Integer                                                *
' *  Dim userInfo As New Collection                                      *
' *                                                                      *
Public Sub HDBbuild1(instruct, Heading, LabelList, defaultInfo, _
                     typeList, indxList, strtIndex, colmnList, nRows, _
                     userInfo)
' *                                                                      *
' *  PURPOSE:  BUILD A CUSTOMIZABLE HORIZONTAL DIALOG BOX WITH THE       *
' *            ABILITY TO SPECIFY DEFAULT VALUES FOR EACH COLUMN ON      *
' *            EACH ROW                                                  *
' *                                                                      *
' *  GIVEN:    instruct    = message box instruction                     *
' *            Heading     = message box heading (title)                 *
' *            labelList   = list of column labels to be displayed       *
' *            defaultInfo = list of default values for each column      *
' *            typeList    = list of item data types for each column     *
' *                          1 = data line, 2 = combo box                *
' *            indxList    = default value for data line data types      *
' *                          or the index position into a combo box      *
' *                          data type denoting the default, values      *
' *                          start at 1 for the index position, enter    *
' *                          NULL to use the global column value,        *
' *                          defaultInfo, for data line data types       *
' *            strtIndex   = pointer into indxList where to begin        *
' *                          extracting data, starting at 1              *
' *            colmnList   = list of column widths                       *
' *             nRows        = number of rows in the dialog box          *
' *                                                                      *
' *   RETURN:    userInfo     = list of user responses for data items     *
' *                                                                      *
' *   NOTE:       (a) The indxList list is a sequential collection whose *
' *                   value is either: (1) the index position into a     *
' *                   combo box data type for the default value or is    *
' *                   (2) the default value for a data line data type.   *
' *                   If the data type at a position is a data line and  *
' *                   the global column default value (defaultInfo) is   *
' *                   to be used, enter NULL.  The size of indxList is   *
' *                   equal to the total number of rows to be displayed  *
' *                 times the number of columns                          *
' *                 Structure of indxList is:                            *
' *                   Item 1: default value or index for column 1, row 1 *
' *                   Item 2: default value or index for column 2, row 1 *
' *                   Item 3: default value or index for column 3, row 1 *
' *                   Item 4: default value or index for column 1, row 2 *
' *                   Item 5: default value or index for column 2, row 2 *
' *                   Item 6: default value or index for column 3, row 2 *
' *                   Item 7: default value or index for column 1, row 3 *
' *                   Item 8: default value or index for column 2, row 3 *
' *                   Item 9: default value or index for column 3, row 3 *
' *               (b) strtIndex is used when multiple forms are to be    *
' *                   shown one after the other, if only one form is     *
' *                   desired strtIndex is 1, otherwise, strtIndex will  *
' *                   denote the starting index into indxList where data *
' *                 is to be extracted                                   *
' *                                                                      *
' *   Dim instruct, Heading As String                                    *
' *   Dim labelList As New Collection                                    *
' *   Dim defaultInfo As New Collection                                  *
' *   Dim typeList As New Collection                                     *
' *   Dim indxList As New Collection                                     *
' *   Dim strtIndex As Long                                              *
' *   Dim colmnList As New Collection                                    *
' *   Dim nRows As Integer                                               *
' *   Dim userInfo As New Collection                                     *
' *                                                                      *
Public  Function  icasinan(ANGLEX)  As  Double
' *                                                                      *
' *   PURPOSE:  COMPUTE THE ARCSIN OF A VALUE                            *
' *                                                                      *
' *   GIVEN:     angleX    = sin of an angle in radians                  *
' *                                                                      *
' *   RETURN:    icasinan = angle in radians whose sin is angleX         *
' *                                                                      *
' *   Dim angleX, icasinan As Double                                     *
' *                                                                      *
Public  Function  icatan(D)  As  Double
' *                                                                      *
' *   PURPOSE:  COMPUTE THE ARC TANGENT OF A NUMBER                      *
' *                                                                      *
' *   GIVEN:     D       = a number                                      *
' *                                                                      *
' *   RETURN:    icatan = arc tangent of a number in radians             *
' *                                                                      *
' *   Dim D, icatan As Double                                            *
' *                                                                      *
Public  Function  iccomdis(X1,  Y1,  X2,  Y2)  As  Double
' *                                                                      *
' *   PURPOSE:  TO COMPUTE THE DISTANCE BETWEEN TWO POINTS               *
' *                                                                      *
' *   GIVEN:     X1,Y1     = coordinates of the first point              *
' *              X2,Y2     = coordinates of the second point             *
' *                                                                      *
' *   RETURN:    iccomdis = distance between the two points              *
' *                                                                      *
' *   Dim X1, Y1, X2, Y2, iccomdis As Double                             *
' *                                                                       *
Public  Sub  iccomppt(XCORD,  YCORD,  X2,  Y2,  XCRD2,  YCRD2,  noFnd)
' *                                                                       *
' *   PURPOSE:   TO CHECK IF A POINT IS WITHIN A TOLERANCE, THAT VARIES  *
' *               BASED UPON THE DISPLAY OF THE VIEW, OF ANOTHER POINT    *
' *                                                                       *
' *   GIVEN:     XCORD,YCORD = coordinates of point to be checked         *
' *              X2,Y2       = coordinates of the base point              *
' *                                                                       *
' *   RETURN:    XCRD2,YCRD2 = input values if NOFND = 0                  *
' *                          = X2,Y2 values if NOFND = 1                  *
' *              NOFND       = 0 : no match was found                     *
' *                          = 1 : match was found within the point       *
' *                                snapping tolerance.                    *
' *                                                                       *
' *   Dim XCORD, YCORD, X2, Y2, XCRD2, YCRD2 As Double                    *
' *   Dim NOFND As Integer                                                *
' *                                                                       *
Public  Function  icdegrad(angle)  As  Double
' *                                                                       *
' *   PURPOSE:   TO CONVERT FROM DEGREES TO RADIANS                       *
' *                                                                       *
' *   GIVEN:     ANGLE     = angle in degrees (decimal)                   *
' *                                                                       *
' *   RETURN:    icdegrad = angle in radians                              *
' *                                                                       *
' *   Dim ANGLE As Double                                                 *
' *   Dim icdegrad As Double                                              *
' *                                                                       *
Public  Sub  icforce(PTN1,  PTE1,  PTN2,  PTE2,  D,  AZ)
' *                                                                       *
' *   PURPOSE:   INVERSE FROM POINT 1 TO POINT 2                          *
' *                                                                       *
' *   GIVEN:     PTN1,PTE1 = north-east coordinates of point 1            *
' *              PTN2,PTE2 = north-east coordinates of point 2            *
' *                                                                       *
' *   RETURN:    D          = distance from point 1 to point 2            *
' *              az         = azimuth (radians) from point 1 to 2         *
' *                                                                       *
' *   Dim PTN1, PTE1, PTN2, PTE2, D, AZ As Double                         *
' *                                                                       *
Public  Function  icmakdir(X1,  Y1,  X2,  Y2)  As  Double
' *                                                                       *
' *   PURPOSE:   COMPUTE THE CARTESIAN DIRECTION OF TWO POINTS            *
' *                                                                       *
' *   GIVEN:     X1,Y1     = coordinates of the first point               *
' *              X2,Y2     = coordinates of the second point              *
' *                                                                       *
' *   RETURN:    icmakdir = direction of the two points in radians        *
' *                                                                       *
' *   Dim X1, Y1, X2, Y2, icmakdir As Double                              *
' *                                                                       *
Public  Function  icraddeg(ANGLE)  As  Double
' *                                                                       *
' *   PURPOSE:   TO CONVERT FROM RADIANS TO DEGREES                       *
' *                                                                       *
' *   GIVEN:     ANGLE     = angle in radians                             *
' *                                                                       *
' *   RETURN:    icraddeg = angle in degrees (decimal)                    *
' *                                                                       *
' *   Dim ANGLE As Double                                                 *
' *   Dim icraddeg As Double                                              *
' *                                                                       *
Public  Sub  LoadVBAcode()
' *                                                                       *
' *   PURPOSE:   LOAD VBA COMPONENTS FROM A DIRECTORY INTO THE CURRENT   *
' *              ACTIVE PROJECT                                           *
' *                                                                       *
' *   GIVEN:     nothing                                                  *
' *                                                                    *
' *  RETURN:   nothing                                                 *
' *                                                                    *
' *   NOTE:      (a) The directory is a hardcoded path stored in aDIR, *
' *              (b) The Avenue Wraps: avGetWorkDir,                   *
' *                                    avListFiles,                    *
' *                                    avSetWorkDir, and               *
' *                                    CreateList                      *
' *                  must appear in the project file in order for this *
' *                  macro to execute                                  *
' *                                                                    *
Public  Function  MakeTextElement(sText,  DX,  DY,  dAngle,  _
                              pTextSymbol As ITextSymbol) As ITextElement
' *                                                                    *
' *  PURPOSE:  TO CREATE A TEXT ELEMENT                                *
' *                                                                    *
' *  GIVEN:     sText            = text string to appear               *
' *             dX,dY            = low left corner coordinates         *
' *             dAngle           = text angle of inclination (degrees) *
' *             pTextSymbol      = text symbol reflecting font, size   *
' *                                and color                           *
' *                                                                    *
' *   RETURN:    MakeTextElement = text element representing the text  *
' *                                                                    *
' *  Dim sText As String                                               *
' *  Dim dX, dY, dAngle As Double                                      *
' *  Dim pTextSymbol As ITextSymbol                                    *
' *  Dim MakeTextElement As ITextElement                               *
' *                                                                    *
Public  Function  MakeTextSymbol(strFont,  dFontSize,  _
                              iItalic, iBold, iColor) As ITextSymbol
' *                                                                    *
' *  PURPOSE:  TO CREATE A TEXT SYMBOL                                 *
' *                                                                    *
' *  GIVEN:     strFont         = text font                            *
' *             dFontSize       = font size                            *
' *             iItalic         = font style (1 = normal, 2 = italic)  *
' *             iBold           = font style (1 = normal, 3 = bold)    *
' *             iColor          = RGB color index value                *
' *                                                                    *
' *   RETURN:    MakeTextSymbol = text symbol representing the text    *
' *                                                                    *
' *  Dim strFont As String                                             *
' *  Dim dFontSize As Double                                           *
' *  Dim iItalic, iBold As Integer                                     *
' *  Dim iColor As Long                                                *
' *  Dim MakeTextSymbol As ITextSymbol                                 *
' *                                                                    *
Public  Sub  RunProgress(xyzRec,  totRecs,  aMessage)
' *                                                                    *
' *   PURPOSE:   Initialize, report on, and terminate the reporting of *
' *              the progress for a processing operation.              *
' *                                                                    *
' *   GIVEN:     xyzRec    = the current unit of measure of progress   *
' *                          = 0: initiate the progress report phase   *
' *                          > 0: report on the progress               *
' *                          < 0: terminate the progress report phase  *
' *              totRecs   = the total unit of measure of progress     *
' *              aMessage = identification of the progress reporting   *
' *                                                                    *
' *  RETURN:   nothing                                                 *
' *                                                                    *
' *   NOTE:      The progress bar can appear in one of two forms, the  *
' *              first is when no stop button is displayed. In this    *
' *              form the progress bar and message appear in the status *
' *              bar area and remain visible until the progress bar is *
' *              terminated. In the second form, the progress bar will *
' *              appear in the middle of the display in a dialog box   *
' *              containing the cancel button. Selecting the cancel    *
' *              button will set the global variable ugpProCancel to be *
' *              TRUE.  By testing the value of ugpProCancel, the user  *
' *              can detect if the operation should be canceled or not. *
' *              In addition, the ugpProDesc variable can be used to    *
' *              display additional information about the operation.    *
' *                                                                      *
' *  Dim xyzRec, totRecs As Long                                        *
' *  Dim aMessage As Variant                                            *
' *                                                                      *
Public  Sub  SetViewSnapTol(theView,  xP,  yP,  _
                             viewRect, thePoint, difxxx, difzzz, difwww)
' *                                                                      *
' *   PURPOSE:   SCRIPT TO SET THE SNAP TOLERANCE FOR THE CURRENT VIEW  *
' *                                                                      *
' *  GIVEN:     theView  = the current active view                      *
' *              xP       = x coordinate of given point                 *
' *              yP       = y coordinate of given point                 *
' *                                                                      *
' *   RETURN:    viewRect = the width of the current visible display    *
' *              thePoint = point in the projected coordinate system,   *
' *                          will be xP,yP if no projection applied, if *
' *                          a projection is applied will be different  *
' *                        from xP,yP                                   *
' *              difxxx    = tolerance as a percentage of the view width *
' *                          based upon user-defined value for ugsnapTol *
' *              difzzz    = smaller tolerance (difxxx * 0.1)           *
' *              difwww    = tolerance which will be:                   *
' *                          (a) the same as difxxx if the tolerance is *
' *                              defined as a percentage                *
' *                              (ugsnapTolMode = "P"), or              *
' *                          (b) equal to the absolute tolerance value  *
' *                              (ugsnapTol), which is converted into   *
' *                              the projected environment, if the      *
' *                              tolerance is defined to be absolute    *
' *                              (ugsnapTolMode = "A")                  *
' *                                                                      *
' *  Dim theView As Variant                                             *
' *  Dim xP, yP As Double                                               *
' *  Dim viewRect, difxxx, difzzz, difwww As Double                     *
' *  Dim thePoint As IPoint                                             *
' *                                                                      *
Public  Sub  ShapeProjectSHP(theFeature  As  IFeature,  _
                             theShape As IGeometry, _
                             ipmode, _
                             theNewFeature As IFeature, _
                             theNewGeometry As IGeometry)
' *                                                                      *
' *   PURPOSE:   SCRIPT TO ALTER A SHAPE BASED UPON THE VIEW PROJECTION *
' *                                                                      *
' *  GIVEN:     theFeature      = the feature to be projected           *
' *              theShape        = the shape to be projected            *
' *              ipmode          = the mode of operation (see below)    *
' *                                 0 : Get the geodetic projection of  *
' *                                     theShape                        *
' *                                 1 : Get the geodetic unprojection of *
' *                                     theShape                        *
' *                                -1 : Check if a special feature type *
' *                                     is to be generated              *
' *                                10 : Check if a special feature type *
' *                                     is to be generated and convert  *
' *                                     the given feature accordingly   *
' *                                11 : Convert PolyLine into PolyLineM *
' *                                12 : Convert PolyLine into PolyLineZ *
' *                                13 : Convert Polygon into PolygonM   *
' *                                14 : Convert Polygon into PolygonZ   *
' *                                15 : Convert Point into PointM       *
' *                                16 : Convert Point into PointZ       *
' *                                17 : Convert Point into MultiPointM  *
' *                                18 : Convert Point into MultiPointZ  *
' *                                                                    *
' *  RETURN:    theNewFeature   = the new feature as projected         *
' *              theNewGeometry = the new shape as projected           *
' *                                                                    *
' *  Dim theFeature As IFeature                                        *
' *  Dim theShape As IGeometry                                         *
' *  Dim ipmode As Integer                                             *
' *  Dim theNewFeature As IFeature, theNewGeometry As IGeometry        *
' *                                                                    *
Public Sub ShapeProjectVAL(theValue, ipmode, theNewValue)
' *                                                                    *
' *  PURPOSE:  CONVERT A VALUE FROM DISTANCE UNITS INTO MAP UNITS OR   *
' *            MAP UNITS INTO DISTANCE UNITS                           *
' *                                                                    *
' *  GIVEN:     theValue     = the value to be converted               *
' *              ipmode       = the mode of operation (see below)      *
' *                             2  : Convert a Distance value from     *
' *                                  Distance Units into Map Units     *
' *                             3  : Convert a Distance value from     *
' *                                  Map Units into Distance Units     *
' *                             4  : Convert an Area value from        *
' *                                  Map Units into Distance Units     *
' *                             5  : Convert an Area value from        *
' *                                  Distance Units into Map Units     *
' *                             92 : Convert a Distance value from     *
' *                                  Distance Units into Map Units     *
' *                                  like ipmode = 2 but will ignore the *
' *                                  projection that is assigned to the  *
' *                                  view, as such, the result will be   *
' *                                  in decimal degrees, if the view has *
' *                                  no projection it operates just like *
' *                                ipmode = 2                          *
' *                                                                    *
' *  RETURN:    theNewValue = the converted value                      *
' *                                                                    *
' *  Dim theValue As Variant, ipmode As Integer                        *
' *  Dim theNewValue As Variant                                        *
' *                                                                    *
Public Sub SortFourArrays(array1, array2, array3, array4, aMssg, _
                            anOrder)
' *                                                                    *
' *  PURPOSE:  SCRIPT TO SORT UP TO FOUR DIFFERENT ARRAYS, SORTING     *
' *             THE OTHER ARRAYS BASED UPON THE SORT OF THE FIRST ONE  *
' *                                                                    *
' *  GIVEN:     array1    = first array of items to be sorted          *
' *              array2    = second array of items to be sorted        *
' *              array3    = third array of items to be sorted         *
' *              array4    = fourth array of items to be sorted        *
' *             aMssg     = progress bar message                       *
' *             anOrder   = the sort order as a Boolean                *
' *                          True = ascending, False = Descending      *
' *                                                                    *
' *  RETURN:    nothing                                                *
' *                                                                    *
' *  NOTE:      (a) The order of the arrays passed in are changed by   *
' *                  this script to reflect the effects of the sort    *
' *             (b) If only one array is to be sorted the array,       *
' *                  array2, array3, array4 can be passed in as NULL   *
' *              (c) If NULL is specified for aMssg, no progress bar   *
' *                  will be displayed                                 *
' *              (d) The arrays can contain string or numeric data but *
' *                  no objects                                        *
' *              (e) All elements in array1 will be sorted, as well as,*
' *                  array2, array3, and array4, if specified          *
' *                                                                    *
' *  Dim array1(), array2(), array3(), array4()                        *
' *  Dim aMssg As Variant, anOrder As Boolean                          *
' *                                                                    *
```

```
Public Sub SortTwoArrays(array1, array2, aMssg, anOrder)
' *                                                                      *
' *   PURPOSE:   SCRIPT TO SORT UP TO TWO DIFFERENT ARRAYS, SORTING THE  *
' *               SECOND ARRAY BASED UPON THE SORT OF THE FIRST ARRAY    *
' *                                                                      *
' *  GIVEN:     array1    = first array of items to be sorted            *
' *              array2    = second array of items to be sorted          *
' *             aMssg     = progress bar message                         *
' *             anOrder   = the sort order as a Boolean                  *
' *                            True = ascending, False = Descending      *
' *                                                                      *
' *  RETURN:   nothing                                                   *
' *                                                                      *
' *   NOTE:       (a) The order of the arrays passed in are changed by   *
' *                    this script to reflect the effects of the sort    *
' *              (b) If only one array is to be sorted the array,        *
' *                   array2 can be passed in as NULL                    *
' *               (c) If NULL is specified for aMssg, no progress bar    *
' *                  will be displayed                                   *
' *               (d) The arrays can contain string or numeric data but  *
' *                  no objects                                          *
' *               (e) All elements in array1 will be sorted, as well as, *
' *                  array2, if specified                                *
' *                                                                      *
' *  Dim array1(), array2()                                              *
' *  Dim aMssg As Variant, anOrder As Boolean                            *
' *                                                                      *
Public Sub SortTwoLists(list1, list2, aMssg, anOrder)
' *                                                                      *
' *   PURPOSE:   SCRIPT TO SORT UP TO TWO DIFFERENT LISTS, SORTING THE   *
' *               SECOND LIST BASED UPON THE SORT OF THE FIRST LIST      *
' *                                                                      *
' *  GIVEN:     list1    = first list of items to be sorted              *
' *              list2    = second list of items to be sorted            *
' *             aMssg    = progress bar message                          *
' *             anOrder = the sort order as a Boolean                    *
' *                           True = ascending, False = Descending       *
' *                                                                      *
' *  RETURN:   nothing                                                   *
' *                                                                      *
' *   NOTE:       (a) The order of the lists passed in are changed by    *
' *                    this script to reflect the effects of the sort    *
' *              (b) If only one list is to be sorted the collection     *
' *                    list2 can be an empty list or passed in as NOTHING *
' *               (c) If NULL is specified for aMssg, no progress bar    *
' *                  will be displayed                                   *
' *                                                                      *
' *   Dim list1 As New Collection, list2 As New Collection               *
' *  Dim aMssg As Variant, anOrder As Boolean                            *
' *                                                                      *
Public Sub VCBbuild(instruct, Heading, LabelList, defaultInfo, _
                       typeList, userInfo)
' *                                                                      *
' *   PURPOSE:   BUILD A CUSTOMIZABLE VERTICAL CHOICE DIALOG BOX         *
' *                                                                      *
' *  GIVEN:     instruct     = message box instruction                   *
' *              Heading      = message box heading (title)              *
' *               labelList    = list of labels for data items (not used, *
' *                               can be NOTHING or an empty list        *
' *               defaultInfo = list of choices (options) user can pick  *
' *              typeList     = type of data item to be displayed (not   *
' *                               used, can be NOTHING or an empty list) *
' *                                                                      *
' *   RETURN:    userInfo     = list containing the selected option      *
' *                                                                      *
' *   NOTE:       Once the user selects an option from the message box,  *
' *              the message box is closed and the option that was       *
' *               selected is passed back. If the message box is closed  *
' *               by the user, userInfo is passed back as an empty list  *
' *                                                                        *
' *  Dim instruct, Heading As String                                       *
' *   Dim labelList As New Collection, defaultInfo As New Collection       *
' *  Dim typeList As New Collection                                        *
' *  Dim userInfo As New Collection                                        *
' *                                                                        *
Public Sub VDBbuild(instruct, Heading, LabelList, defaultInfo, _
                      typeList, userInfo)
' *                                                                        *
' *  PURPOSE:  BUILD A CUSTOMIZABLE VERTICAL DIALOG BOX                    *
' *                                                                        *
' *  GIVEN:    instruct    = message box instruction                       *
' *             Heading     = message box heading (title)                  *
' *             labelList   = list of labels for data items                *
' *              defaultInfo = list of default values for data items       *
' *             typeList    = type of data item to be displayed            *
' *                            1 = data line, 2 = combo box,               *
' *                            3 = text box with multiselect               *
' *  RETURN:   userInfo    = list of user responses for data items         *
' *                                                                        *
' *  NOTE:      Only use a text box control when a single data item        *
' *              is to be displayed such as done with avMsgBoxMultiList    *
' *  Dim instruct, Heading As String                                       *
' *   Dim labelList As New Collection, defaultInfo As New Collection       *
' *  Dim typeList As New Collection                                        *
' *  Dim userInfo As New Collection                                        *
' *                                                                        *
Public Sub VDBbuild2(instruct, Heading, labelList, defaultInfo, _
                       typeList, userInfo)
' *                                                                        *
' *  PURPOSE:  BUILD A CUSTOMIZABLE VERTICAL DIALOG BOX WITH A BACK        *
' *            BUTTON                                                      *
' *                                                                        *
' *  GIVEN:    instruct    = message box instruction                       *
' *             Heading     = message box heading (title)                  *
' *             labelList   = list of labels for data items                *
' *              defaultInfo = list of default values for data items       *
' *             typeList    = type of data item to be displayed            *
' *                            1 = data line, 2 = combo box                *
' *                                                                        *
' *  RETURN:   userInfo    = list of user responses for data items         *
' *                                                                        *
' *  Dim instruct, Heading As String                                       *
' *   Dim labelList As New Collection, defaultInfo As New Collection       *
' *  Dim typeList As New Collection                                        *
' *  Dim userInfo As New Collection                                        *
```