Appendix A: Implicit linking in PREMIS

The method of implicit linking was used in the precedent system PREMIS as a means of connecting alphanumeric information in a relational database to the graphic objects in an indexed graphical object library. This was a very convenient means of connecting diverse sources of information together. Consider Code Fragment 6 below that contains the ASCII representation of a simple rectangular graphic entity in PREMIS. The `subject` label indicates that the subject under consideration is related to `SPACE`. The `object` label indicates that the space is called `FS28:B7:F0:E2`. This is essentially a hierarchy that indicates that the graphical object describes the shape of a room `E2` that occurs on floor `F0` in a building called `B7` and a facility identified as `FS28`. The `:` separates the different parts of the object name or facets. It is interesting to note that this type of hierarchy fits naturally into the modern XML hierarchical paradigm. At the time when this format was used the processing speed of computers were such that raw ASCII code would have been inefficient. The fragment below was compiled into an efficient binary format and indexed with a highly optimised hashing procedure. The efficiency of current computers makes the use of structured ASCII code such as HTML, XML and the code fragment below feasible. The use of ASCII coding huge advantages such as:

- Very easy to read and understand
- Non-proprietary neutral knowledge formats that can be interpreted by any compliant software applications
- Very long life of data that can easily outlive the application that originally created it

```
. . .
subject
SPACE
object
FS28:B7:F0:E2
Hook
43.814300 –149.229000 0.000000
extents
1
4
40.965500 –149.532200 0.000000
44.323400 –152.048800 0.000000
46.663200 –148.926800 0.000000
43.305300 –146.410200 0.000000
drawing
E2
Scale
1.000000
rotation
0.000000 1.000000
world
0.000000 0.000000 0.000000
. . .
```

Code Fragment 6: Structure of a typical PREMIS graphical record

To connect a relational database record to the graphical record it is only necessary to create a database table record with four not null keys in a database such as Oracle or SQLServer. Relational database technology ensures that the combination of the key fields will always be unique. Two further constraints were placed on the database records:

The key fields must only contain uppercase characters
All key fields must have values (not null)

If during a query a user wants to display all graphical records related to the database a very simple SQL statement could be used such as:

```
SELECT
Space.SiteId||’:’||Space.BuildingId||’:’||Space.FloorId||’:’||Space.S
paceId FROM Space WHERE SiteId = ‘FS28’
```

By means of the concatenation of the key fields in abovementioned statement the graphical and alphanumeric records are logically related. This method is known as *implicit linking* because graphical and alphanumeric records are related by virtue of the similarity in the names. This offers the following important advantages:

- Data from diverse sources can easily be related together
- One alphanumeric relational database record can have multiple graphical representations ranging from outline to highly detailed
- Different operators (knowledge workers) can create the information knowing that it is logically related
- Information can conveniently be exported and imported from diverse distributed environments

The disadvantage of this method is that classification system must still be agreed on beforehand. Facilities managers have to decide what the codes should be and the graphical records must be structured in a similar way. This can nowadays be overcome by using a *Global Unique Identifier* (GUID) such as used in ActiveX controls. This provides the ultimate in globally unique codes. The only drawback of a GUID is that the code is non-mnemonic of nature making it difficult to know on face value what it relates to.

Appendix B: PREMIS search criteria definition

The Following logic has been used for the search area definition in PREMIS.

Consider polygon $p_1 \ldots p_n$ and an arbitrary polygon *s*.

Define a function *T (s,i)* = { *O* if *s* is outside p_i; *I* if *s* is inside p_i; *C* if *s* if s crosses p_i

Union

Inside:	*i* 0 ⌉*n : T(s,i) = I*
Inside crossing:	*i* 0 ⌉*n : T(s,i)* 0 {*I,C*}
Crossing:	*i* 0 ⌉*n : T(s,i) = C* ϖ *i* 0 ⌉*n : T(s,i)* 0 {*O,C*}

Intersection

Inside:	*i* 0 ⌉*n : T(s,i) = I*
Inside crossing:	*i* 0 ⌉*n : T(s,i)* 0 {*I,C*}
Crossing:	*i* 0 ⌉*n : T(s,i)* 0 {*I,C*} ϖ *i* 0 ⌉*n : T(s,i) = C*

Excluded intersection

Inside:	*i* 0 ⌉*n : T(s,i) = I* ϖ *i* 0 ⌉*n : T(s,i) = O*
Inside crossing:	*i* 0 ⌉*n : T(s,i)* 0 {*I,C*} ϖ *i* 0 ⌉*n : T(s,i)* 0 {*O,C*}
Crossing:	*i* 0 ⌉*n : T(s,i) = C* ϖ (*i* 0 ⌉*n : T(s,i)* 0 {*I,C*} ω *i* 0 ⌉*n : T(s,i)* 0 {*O,C*})

Appendix C: Interface an ActiveX control to an Excel spreadsheet

```
Private Sub ArgosAB_GotFocus()
    Worksheets("Sheet1").Range("A1:A10").Value = ArgosAB.GrossArea
    txtArgos.Text = ArgosAB.GrossArea
End Sub

Private Sub ArgosAB_LostFocus()
    Worksheets("Sheet1").Range("A1:A10").Value = ""
    txtArgos.Text = "RESET TO EMPTY"
End Sub
```

Appendix D: XSL stylesheet to convert XML into VML for web page display

```
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:v="urn:schemas-microsoft-com:vml">

   <xsl:template match="MicroGDS">
      <xsl:variable name="LX">
         <xsl:for-each select="Layer/Extent">
            <xsl:sort select="number(@LX)" data-type="number" order="ascending"/>
            <xsl:if test="position()=1">
               <xsl:value-of select="number(@LX)"/>
            </xsl:if>
         </xsl:for-each>
      </xsl:variable>

      <xsl:variable name="LY">
         <xsl:for-each select="Layer/Extent">
            <xsl:sort select="number(@LY)" data-type="number" order="ascending"/>
            <xsl:if test="position()=1">
               <xsl:value-of select="number(@LY)"/>
            </xsl:if>
         </xsl:for-each>
      </xsl:variable>

      <xsl:variable name="HX">
         <xsl:for-each select="Layer/Extent">
            <xsl:sort select="number(@HX)" data-type="number" order="descending"/>
            <xsl:if test="position()=1">
               <xsl:value-of select="number(@HX)"/>
            </xsl:if>
         </xsl:for-each>
      </xsl:variable>

      <xsl:variable name="HY">
         <xsl:for-each select="Layer/Extent">
            <xsl:sort select="number(@HY)" data-type="number" order="descending"/>
            <xsl:if test="position()=1">
               <xsl:value-of select="number(@HY)"/>
            </xsl:if>
         </xsl:for-each>
      </xsl:variable>

      <xsl:variable name="SX">
         <xsl:value-of select="$HX - $LX"/>
      </xsl:variable>

      <xsl:variable name="SY">
         <xsl:value-of select="$HY - $LY"/>
      </xsl:variable>

      <xsl:variable name="WW">
         800
      </xsl:variable>

      <xsl:variable name="HH">
         <xsl:value-of select="$WW * $SY div $SX"/>
      </xsl:variable>

      <v:group
         style="position: absolute; margin-left: 10px; margin-top: 10px; width:
{$WW}px; height: {$HH}px;"
         coordsize="{$SX},{$SY}"
         coordorigin="{$LX},{$LY}"
      >

      <!--frame-->

      <v:polyline
         points="{$LX},{$LY},{$HX},{$LY},{$HX},{$HY},{$LX},{$HY},{$LX},{$LY}"
      />

      <!--layer-->
```

```
<xsl:for-each select="Window/Phase">
   <xsl:if test="not(@State[.='Invisible'])">

      <xsl:variable name="LinkNo">
         <xsl:value-of select="@Layer"/>
      </xsl:variable>

      <xsl:for-each select="../../Layer">

         <xsl:if test="@LinkNumber[.= $LinkNo]">

            <!--lines-->
            <xsl:for-each select="OCD/Object/LinePrimitive">

               <v:polyline filled="false">
                  <xsl:attribute name="points">
                     <xsl:for-each select="Polyline/Point">
                        <xsl:value-of select="number(@X)"/>,
                        <xsl:value-of select="$HY + $LY - number(@Y)"/>,
                     </xsl:for-each>
                  </xsl:attribute>
               </v:polyline>

            </xsl:for-each>
            <!--end line-->

            <!--text-->
            <v:shapetype id="TextPrim" coordsize="21600,21600"
               path="m0,-14400l21600,-14400e">

            <v:path textpathok="t" />
            <v:textpath on="t" fitshape="t" xscale="t"/>
            </v:shapetype>


            <xsl:for-each select="OCD/Object/TextPrimitive">

               <!--extent-->
               <xsl:variable name="CharLX">
                  <xsl:for-each select="Extent">
                     <xsl:value-of select="number(@LX)"/>
                  </xsl:for-each>
               </xsl:variable>
               <xsl:variable name="CharLY">
                  <xsl:for-each select="Extent">
                     <xsl:value-of select="$HY + $LY - number(@LY)"/>
                  </xsl:for-each>
               </xsl:variable>
               <xsl:variable name="CharHX">
                  <xsl:for-each select="Extent">
                     <xsl:value-of select="number(@HX)"/>
                  </xsl:for-each>
               </xsl:variable>
               <xsl:variable name="CharHY">
                  <xsl:for-each select="Extent">
                     <xsl:value-of select="$HY + $LY - number(@HY)"/>
                  </xsl:for-each>
               </xsl:variable>
               <xsl:variable name="CharEX">
                  <xsl:value-of select="$CharHX - $CharLX"/>
               </xsl:variable>
               <xsl:variable name="CharEY">
                  <xsl:value-of select="$CharLY - $CharHY"/>
               </xsl:variable>

               <!--Char String-->
               <xsl:variable name="CharStringExpanded">
                  <xsl:value-of select="ExpandedText"/>
               </xsl:variable>
               <xsl:variable name="CharString">
                  <xsl:choose>
                     <xsl:when test="$CharStringExpanded=''">
                        <xsl:value-of select="DefinitionText"/>
                     </xsl:when>
                     <xsl:otherwise>
                        <xsl:value-of select="$CharStringExpanded"/>
                     </xsl:otherwise>
                   </xsl:choose>
               </xsl:variable>

               <!--CharStyle-->
               <xsl:variable name="CharFontType">
                  <xsl:value-of select="@Charstyle"/>
               </xsl:variable>
               <xsl:variable name="CharFontFamily">
                  <xsl:choose>
                     <xsl:when test="$CharFontType=''">
                        Times New Roman
                     </xsl:when>
                     <xsl:when test="$CharFontType='DEFAULT'">
                        Times New Roman
                     </xsl:when>
                     <xsl:otherwise>
                        <xsl:for-each select="../../../../Styles/TTCharstyle">
                           <xsl:if test="@Name[.= $CharFontType]">
                              <xsl:value-of select="@FontName"/>
                           </xsl:if>
                        </xsl:for-each>
                     </xsl:otherwise>
                  </xsl:choose>
               </xsl:variable>

               <!--Italic-->
               <xsl:variable name="CharItalic">
                  <xsl:choose>
                     <xsl:when test="$CharFontType=''">
                        normal
                     </xsl:when>
                     <xsl:when test="$CharFontType='DEFAULT'">
                        normal
                     </xsl:when>
                     <xsl:otherwise>
                        <xsl:for-each select="../../../../Styles/TTCharstyle">
                           <xsl:if test="@Name[.=$CharFontType]">
                              <xsl:choose>
                                 <xsl:when test="@Italic='true'">
                                    italic
                                 </xsl:when>
                                 <xsl:otherwise>
                                    normal
                                 </xsl:otherwise>
                              </xsl:choose>
                           </xsl:if>
                        </xsl:for-each>
                     </xsl:otherwise>
                  </xsl:choose>
               </xsl:variable>

               <!--text align-->
               <xsl:variable name="CharJustify">
                  <xsl:value-of select="@Justification"/>
               </xsl:variable>
               <xsl:variable name="CharAlign">
                  <xsl:choose>
                     <xsl:when test="$CharJustify='BL'">
                        left
                     </xsl:when>
                     <xsl:when test="$CharJustify='CL'">
                        left
                     </xsl:when>
                     <xsl:when test="$CharJustify='TL'">
                        left
                     </xsl:when>
                     <xsl:when test="$CharJustify='BC'">
                        center
                     </xsl:when>
                     <xsl:when test="$CharJustify='CC'">
                        center
                     </xsl:when>
                     <xsl:when test="$CharJustify='TC'">
                        center
                     </xsl:when>
                     <xsl:when test="$CharJustify='BR'">
```

```
                              right
                           </xsl:when>
                           <xsl:when test="$CharJustify='CR'">
                              right
                           </xsl:when>
                           <xsl:when test="$CharJustify='TR'">
                              right
                           </xsl:when>
                        </xsl:choose>
                     </xsl:variable>

                     <v:shape type="#TextPrim"
                        style="position:absolute; top: {$CharLY} ; left: {$CharLX};
width: {$CharEX} ;height: {$CharEY};"
                        adj="0" fillcolor="black" strokeweight="1pt">

                     <v:fill method="linear sigma" focus="100%"/>
                     <v:textpath
                        style='font-family : {$CharFontFamily};
                               font-style  : {$CharItalic};
                               font-weight : normal;
                               v-text-align: {$CharAlign};
                               v-text-kern:t'
                         trim="t" fitpath="t" xscale="f" string="{$CharString}"/>
                     </v:shape>

                  </xsl:for-each>
                  <!--end text-->

                </xsl:if>
            </xsl:for-each>

         </xsl:if>
      </xsl:for-each>

      </v:group>
   </xsl:template>
</xsl:stylesheet>
```

Appendix E: Visual Basic code to implement a minimal web browser

```
Option Explicit
Public StartingAddress As String
Dim mbDontNavigateNow As Boolean

Private Sub cboAddress_Click()
    If mbDontNavigateNow Then Exit Sub
    timTimer.Enabled = True
    brwWebBrowser.Navigate cboAddress.Text

End Sub

Private Sub cboAddress_KeyPress(KeyAscii As Integer)
    On Error Resume Next
    If KeyAscii = vbKeyReturn Then
        cboAddress_Click
    End If
End Sub

Private Sub cmdBack_Click()
    timTimer.Enabled = True
    brwWebBrowser.GoBack
End Sub

Private Sub cmdForward_Click()
    timTimer.Enabled = True
    brwWebBrowser.GoForward

End Sub

Private Sub cmdHome_Click()
    timTimer.Enabled = True
    brwWebBrowser.Navigate StartingAddress
End Sub

Private Sub cmdRefresh_Click()
    timTimer.Enabled = True
    brwWebBrowser.Refresh
End Sub

Private Sub cmdSearch_Click()
    timTimer.Enabled = True
    brwWebBrowser.GoSearch
End Sub

Private Sub cmdStop_Click()
    timTimer.Enabled = False
    brwWebBrowser.Stop
    Me.Caption = brwWebBrowser.LocationName
End Sub

Private Sub Form_Load()
    On Error Resume Next
    Me.Show
    Form_Resize

    StartingAddress = "http://conradie/welcome.htm"
    If Len(StartingAddress) > 0 Then
        cboAddress.Text = StartingAddress
        cboAddress.AddItem cboAddress.Text
        timTimer.Enabled = True
        brwWebBrowser.Navigate StartingAddress
    End If

End Sub

Public Sub brwWebBrowser_NavigateComplete(ByVal URL As String)

    Dim i As Integer
    Dim bFound As Boolean
    Me.Caption = brwWebBrowser.LocationName
    For i = 0 To cboAddress.ListCount - 1
        If cboAddress.List(i) = brwWebBrowser.LocationURL Then
            bFound = True
            Exit For
        End If
    Next i
    mbDontNavigateNow = True
    If bFound Then
        cboAddress.RemoveItem i
    End If
    cboAddress.AddItem brwWebBrowser.LocationURL, 0
    cboAddress.ListIndex = 0
    mbDontNavigateNow = False

End Sub

Private Sub Form_Resize()
    cboAddress.Width = Me.ScaleWidth - 100
    brwWebBrowser.Width = Me.ScaleWidth - 100
    brwWebBrowser.Height = Me.ScaleHeight - 200
End Sub

Private Sub timTimer_Timer()
    If brwWebBrowser.Busy = False Then
        timTimer.Enabled = False
            Me.Caption = brwWebBrowser.LocationName
        Else
            Me.Caption = "Working..."
        End If
End Sub
```

Appendix F: Visual Basic code to implement ARGOS intelligent component

```
'Default Property Values:

Const m_def_BackColor = 0
Const m_def_ForeColor = 0
Const m_def_Enabled = 0
Const m_def_BackStyle = 0
Const m_def_BorderStyle = 0
Const m_def_AA_xdim = 1000
Const m_def_AA_ydim = 1000
Const m_def_AA_zdim = 1000
Const m_def_AA_scale = 1
Const m_def_AA_unit = "mm"
Const m_def_AE_construction_area = 1000
Const m_def_AE_cost = 1
Const m_def_AE_durability = 1
Const m_def_AE_energy_use = 1
Const m_def_AE_grossarea = 1
Const m_def_AE_nettarea = 1
Const m_def_AE_rentable_area = 1
Const m_def_AE_shape = 6
Const m_def_AE_volume = 1
Const m_def_AF_function = ""
Const m_def_AT_hearing = ""
Const m_def_AT_internal_sensitivity = ""
Const m_def_AT_recognition = ""
Const m_def_AT_sight = ""
Const m_def_AT_smell = ""
Const m_def_AT_taste = ""
Const m_def_AE_wall_space_ratio = 0.9

'Property Variables:

Dim m_BackColor As Long
Dim m_ForeColor As Long
Dim m_Enabled As Boolean
Dim m_Font As Font
Dim m_BackStyle As Integer
Dim m_BorderStyle As Integer
Dim m_AA_xdim As Double
Dim m_AA_ydim As Double
Dim m_AA_zdim As Double
Dim m_AA_scale As Double
Dim m_AA_unit As String
Dim m_AE_construction_area As Double
Dim m_AE_cost As Currency
Dim m_AE_durability As Double
Dim m_AE_energy_use As Double
Dim m_AE_grossarea As Double
Dim m_AE_nettarea As Double
Dim m_AE_rentable_area As Double
Dim m_AE_shape As Double
Dim m_AE_volume As Double
Dim m_AF_function As String
Dim m_AT_hearing As String
Dim m_AT_internal_sensitivity As String
Dim m_AT_recognition As String
Dim m_AT_sight As String
Dim m_AT_smell As String
Dim m_AT_taste As String
Dim m_AE_wall_space_ratio As Double

'Event Declarations:

Event Click()
Event DblClick()
Event KeyDown(KeyCode As Integer, Shift As Integer)
Event KeyPress(KeyAscii As Integer)
Event KeyUp(KeyCode As Integer, Shift As Integer)
Event MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
Event MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
Event MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=8,0,0,0
Public Property Get BackColor() As Long
    BackColor = m_BackColor
End Property

Public Property Let BackColor(ByVal New_BackColor As Long)
    m_BackColor = New_BackColor
    PropertyChanged "BackColor"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=8,0,0,0
Public Property Get ForeColor() As Long
    ForeColor = m_ForeColor
End Property

Public Property Let ForeColor(ByVal New_ForeColor As Long)
    m_ForeColor = New_ForeColor
    PropertyChanged "ForeColor"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=0,0,0,0
Public Property Get Enabled() As Boolean
    Enabled = m_Enabled
End Property

Public Property Let Enabled(ByVal New_Enabled As Boolean)
    m_Enabled = New_Enabled
    PropertyChanged "Enabled"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=6,0,0,0
Public Property Get Font() As Font
    Set Font = m_Font
End Property

Public Property Set Font(ByVal New_Font As Font)
    Set m_Font = New_Font
    PropertyChanged "Font"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=7,0,0,0
Public Property Get BackStyle() As Integer
    BackStyle = m_BackStyle
End Property

Public Property Let BackStyle(ByVal New_BackStyle As Integer)
    m_BackStyle = New_BackStyle
    PropertyChanged "BackStyle"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=7,0,0,0
Public Property Get BorderStyle() As Integer
    BorderStyle = m_BorderStyle
End Property

Public Property Let BorderStyle(ByVal New_BorderStyle As Integer)
    m_BorderStyle = New_BorderStyle
    PropertyChanged "BorderStyle"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=5
Public Sub Refresh()

End Sub

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=4,0,0,1000
```

```
Public Property Get AA_xdim() As Double
    AA_xdim = m_AA_xdim
End Property

Public Property Let AA_xdim(ByVal New_AA_xdim As Double)
    m_AA_xdim = New_AA_xdim
    PropertyChanged "AA_xdim"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=4,0,0,1000
Public Property Get AA_ydim() As Double
    AA_ydim = m_AA_ydim
End Property

Public Property Let AA_ydim(ByVal New_AA_ydim As Double)
    m_AA_ydim = New_AA_ydim
    PropertyChanged "AA_ydim"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=4,0,0,1000
Public Property Get AA_zdim() As Double
    AA_zdim = m_AA_zdim
End Property

Public Property Let AA_zdim(ByVal New_AA_zdim As Double)
    m_AA_zdim = New_AA_zdim
    PropertyChanged "AA_zdim"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=4,0,0,1
Public Property Get AA_scale() As Double
    AA_scale = m_AA_scale
End Property

Public Property Let AA_scale(ByVal New_AA_scale As Double)
    m_AA_scale = New_AA_scale
    PropertyChanged "AA_scale"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=13,0,0,mm
Public Property Get AA_unit() As String
    AA_unit = m_AA_unit
End Property

Public Property Let AA_unit(ByVal New_AA_unit As String)
    m_AA_unit = New_AA_unit
    PropertyChanged "AA_unit"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MappingInfo=lblDescription,lblDescription,-1,Caption
Public Property Get AA_name() As String
    AA_name = lblDescription.Caption
End Property

Public Property Let AA_name(ByVal New_AA_name As String)
    lblDescription.Caption() = New_AA_name
    PropertyChanged "AA_name"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=4,0,0,1000
Public Property Get AE_construction_area() As Double
    AE_construction_area = m_AE_construction_area
End Property

Public Property Let AE_construction_area(ByVal New_AE_construction_area As Double)
    m_AE_construction_area = New_AE_construction_area
    PropertyChanged "AE_construction_area"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=2,0,0,1
```

```
Public Property Get AE_cost() As Currency
    AE_cost = m_AE_cost
End Property

Public Property Let AE_cost(ByVal New_AE_cost As Currency)
    m_AE_cost = New_AE_cost
    PropertyChanged "AE_cost"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=4,0,0,1
Public Property Get AE_durability() As Double
    AE_durability = m_AE_durability
End Property

Public Property Let AE_durability(ByVal New_AE_durability As Double)
    m_AE_durability = New_AE_durability
    PropertyChanged "AE_durability"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=4,0,0,1
Public Property Get AE_energy_use() As Double
    AE_energy_use = m_AE_energy_use
End Property

Public Property Let AE_energy_use(ByVal New_AE_energy_use As Double)
    m_AE_energy_use = New_AE_energy_use
    PropertyChanged "AE_energy_use"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=4,0,0,1
Public Property Get AE_grossarea() As Double
    AE_grossarea = m_AE_grossarea
End Property

Public Property Let AE_grossarea(ByVal New_AE_grossarea As Double)
    m_AE_grossarea = New_AE_grossarea
    PropertyChanged "AE_grossarea"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=4,0,0,1
Public Property Get AE_nettarea() As Double
    AE_nettarea = m_AE_nettarea
End Property

Public Property Let AE_nettarea(ByVal New_AE_nettarea As Double)
    m_AE_nettarea = New_AE_nettarea
    PropertyChanged "AE_nettarea"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=4,0,0,1
Public Property Get AE_rentable_area() As Double
    AE_rentable_area = m_AE_rentable_area
End Property

Public Property Let AE_rentable_area(ByVal New_AE_rentable_area As Double)
    m_AE_rentable_area = New_AE_rentable_area
    PropertyChanged "AE_rentable_area"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=4,0,0,6
Public Property Get AE_shape() As Double
    AE_shape = m_AE_shape
End Property

Public Property Let AE_shape(ByVal New_AE_shape As Double)
    m_AE_shape = New_AE_shape
    PropertyChanged "AE_shape"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=4,0,0,1
```

```
Public Property Get AE_volume() As Double
    AE_volume = m_AE_volume
End Property

Public Property Let AE_volume(ByVal New_AE_volume As Double)
    m_AE_volume = New_AE_volume
    PropertyChanged "AE_volume"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=13,0,0,
Public Property Get AF_function() As String
    AF_function = m_AF_function
End Property

Public Property Let AF_function(ByVal New_AF_function As String)
    m_AF_function = New_AF_function
    PropertyChanged "AF_function"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=13,0,0,
Public Property Get AT_hearing() As String
    AT_hearing = m_AT_hearing
End Property

Public Property Let AT_hearing(ByVal New_AT_hearing As String)
    m_AT_hearing = New_AT_hearing
    PropertyChanged "AT_hearing"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=13,0,0,
Public Property Get AT_internal_sensitivity() As String
    AT_internal_sensitivity = m_AT_internal_sensitivity
End Property

Public Property Let AT_internal_sensitivity(ByVal New_AT_internal_sensitivity As
String)
    m_AT_internal_sensitivity = New_AT_internal_sensitivity
    PropertyChanged "AT_internal_sensitivity"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=13,0,0,
Public Property Get AT_recognition() As String
    AT_recognition = m_AT_recognition
End Property

Public Property Let AT_recognition(ByVal New_AT_recognition As String)
    m_AT_recognition = New_AT_recognition
    PropertyChanged "AT_recognition"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=13,0,0,
Public Property Get AT_sight() As String
    AT_sight = m_AT_sight
End Property

Public Property Let AT_sight(ByVal New_AT_sight As String)
    m_AT_sight = New_AT_sight
    PropertyChanged "AT_sight"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=13,0,0,
Public Property Get AT_smell() As String
    AT_smell = m_AT_smell
End Property

Public Property Let AT_smell(ByVal New_AT_smell As String)
    m_AT_smell = New_AT_smell
    PropertyChanged "AT_smell"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
```

```
'MemberInfo=13,0,0,
Public Property Get AT_taste() As String
    AT_taste = m_AT_taste
End Property

Public Property Let AT_taste(ByVal New_AT_taste As String)
    m_AT_taste = New_AT_taste
    PropertyChanged "AT_taste"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MappingInfo=txtDescription,txtDescription,-1,Text
Public Property Get AA_description() As String
    AA_description = txtDescription.Text
End Property

Public Property Let AA_description(ByVal New_AA_description As String)
    txtDescription.Text() = New_AA_description
    PropertyChanged "AA_description"
End Property

'WARNING! DO NOT REMOVE OR MODIFY THE FOLLOWING COMMENTED LINES!
'MemberInfo=4,0,0,0.9
Public Property Get AE_wall_space_ratio() As Double
    AE_wall_space_ratio = m_AE_wall_space_ratio
End Property

Public Property Let AE_wall_space_ratio(ByVal New_AE_wall_space_ratio As Double)
    m_AE_wall_space_ratio = New_AE_wall_space_ratio
    PropertyChanged "AE_wall_space_ratio"
End Property

Private Sub cmd2D_3D_Click()
' Toggle the 2D - 3D mode command button
    Dim Isometric_y As Double
    Dim Isometric_x As Double

    Isometric_y = AA_ydim * 0.5
    Isometric_x = AA_ydim * 0.866

    If cmd2D_3D.Caption = "2" Then
        cmd2D_3D.Caption = "3"

        yzSlide.Value = AA_zdim

        UserControl.Height = AA_zdim + Isometric_y
        UserControl.Width = AA_xdim + Isometric_x
        UserControl.Height = AA_zdim + Isometric_y
        UserControl.Width = AA_xdim + Isometric_x

        cmdEnlarge.Top = Isometric_y + 25.1
        lblDescription.Top = Isometric_y + 25.1
        cmdReduce.Top = Isometric_y + 25.1
        cmd2D_3D.Top = Isometric_y + 361.446
        txtDescription.Top = Isometric_y + 361.446
        txtDescription.Height = AA_zdim - 680.72
        yzSlide.Top = Isometric_y + 25.1
        yzSlide.Height = AA_zdim - 149.498

        lblFrom.Top = UserControl.Height - 325.301
        lblCurrent.Top = UserControl.Height - 325.301
        lblTo.Top = UserControl.Height - 325.301
        xSlide.Top = UserControl.Height - 149.598

        shpDesign.Top = Isometric_y
        shpDesign.Height = UserControl.Height - Isometric_y

        ' Make the isometric projection lines visible

        linLine30_1.Visible = True
        linLine30_1.X1 = 0#
        linLine30_1.Y1 = Isometric_y
        linLine30_1.X2 = Isometric_x
        linLine30_1.Y2 = 0#

        linLine30_2.Visible = True
        linLine30_2.X1 = AA_xdim
        linLine30_2.Y1 = Isometric_y
        linLine30_2.X2 = UserControl.Width
        linLine30_2.Y2 = 0#

        linLine0_1.Visible = True
        linLine0_1.X1 = Isometric_x
        linLine0_1.Y1 = 0#
        linLine0_1.X2 = UserControl.Width
        linLine0_1.Y2 = 0#

        linLine90_1.Visible = True
        linLine90_1.X1 = UserControl.Width
        linLine90_1.Y1 = 0#
        linLine90_1.X2 = UserControl.Width
        linLine90_1.Y2 = UserControl.Height - Isometric_y

        linLine30_3.Visible = True
        linLine30_3.X1 = AA_xdim
        linLine30_3.Y1 = UserControl.Height
        linLine30_3.X2 = UserControl.Width
        linLine30_3.Y2 = UserControl.Height - Isometric_y


    Else

        cmd2D_3D.Caption = "2"

        yzSlide.Value = AA_ydim

        UserControl.Height = AA_ydim
        UserControl.Width = AA_xdim
        UserControl.Width = AA_xdim

        cmdEnlarge.Top = 25.1
        lblDescription.Top = 25.1
        cmdReduce.Top = 25.1
        cmd2D_3D.Top = 361.446
        txtDescription.Top = 361.446
        yzSlide.Top = 25.1

        lblFrom.Top = UserControl.Height - 325.301
        lblCurrent.Top = UserControl.Height - 325.301
        lblTo.Top = UserControl.Height - 325.301
        xSlide.Top = UserControl.Height - 149.598
        shpDesign.Top = 0#
        shpDesign.Height = UserControl.Height
        txtDescription.Height = UserControl.Height - 680.72
        yzSlide.Height = UserControl.Height - 149.498
        xSlide.Top = UserControl.Height - 149.498

        ' Set the projection lines invisible

        linLine30_1.Visible = False
        linLine30_2.Visible = False
        linLine30_3.Visible = False
        linLine0_1.Visible = False
        linLine90_1.Visible = False

    End If
End Sub

'Initialize Properties for User Control
Private Sub UserControl_InitProperties()

    m_BackColor = m_def_BackColor
    m_ForeColor = m_def_ForeColor
    m_Enabled = m_def_Enabled
    Set m_Font = Ambient.Font
    m_BackStyle = m_def_BackStyle
    m_BorderStyle = m_def_BorderStyle
    m_AA_xdim = m_def_AA_xdim
    m_AA_ydim = m_def_AA_ydim
    m_AA_zdim = m_def_AA_zdim
    m_AA_scale = m_def_AA_scale
    m_AA_unit = m_def_AA_unit
    m_AE_construction_area = m_def_AE_construction_area
    m_AE_cost = m_def_AE_cost
    m_AE_durability = m_def_AE_durability
    m_AE_energy_use = m_def_AE_energy_use
    m_AE_grossarea = m_def_AE_grossarea
    m_AE_nettarea = m_def_AE_nettarea
    m_AE_rentable_area = m_def_AE_rentable_area
    m_AE_shape = m_def_AE_shape
    m_AE_volume = m_def_AE_volume
    m_AF_function = m_def_AF_function
    m_AT_hearing = m_def_AT_hearing
    m_AT_internal_sensitivity = m_def_AT_internal_sensitivity
    m_AT_recognition = m_def_AT_recognition
    m_AT_sight = m_def_AT_sight
    m_AT_smell = m_def_AT_smell
    m_AT_taste = m_def_AT_taste
    m_AE_wall_space_ratio = m_def_AE_wall_space_ratio

End Sub

'Load property values from storage
Private Sub UserControl_ReadProperties(PropBag As PropertyBag)

    m_BackColor = PropBag.ReadProperty("BackColor", m_def_BackColor)
    m_ForeColor = PropBag.ReadProperty("ForeColor", m_def_ForeColor)
    m_Enabled = PropBag.ReadProperty("Enabled", m_def_Enabled)
    Set m_Font = PropBag.ReadProperty("Font", Ambient.Font)
    m_BackStyle = PropBag.ReadProperty("BackStyle", m_def_BackStyle)
    m_BorderStyle = PropBag.ReadProperty("BorderStyle", m_def_BorderStyle)
    m_AA_xdim = PropBag.ReadProperty("AA_xdim", m_def_AA_xdim)
    m_AA_ydim = PropBag.ReadProperty("AA_ydim", m_def_AA_ydim)
    m_AA_zdim = PropBag.ReadProperty("AA_zdim", m_def_AA_zdim)
    m_AA_scale = PropBag.ReadProperty("AA_scale", m_def_AA_scale)
    m_AA_unit = PropBag.ReadProperty("AA_unit", m_def_AA_unit)
    lblDescription.Caption = PropBag.ReadProperty("AA_name", "A")
    m_AE_construction_area = PropBag.ReadProperty("AE_construction_area",
m_def_AE_construction_area)
    m_AE_cost = PropBag.ReadProperty("AE_cost", m_def_AE_cost)
    m_AE_durability = PropBag.ReadProperty("AE_durability", m_def_AE_durability)
    m_AE_energy_use = PropBag.ReadProperty("AE_energy_use", m_def_AE_energy_use)
    m_AE_grossarea = PropBag.ReadProperty("AE_grossarea", m_def_AE_grossarea)
    m_AE_nettarea = PropBag.ReadProperty("AE_nettarea", m_def_AE_nettarea)
    m_AE_rentable_area = PropBag.ReadProperty("AE_rentable_area",
m_def_AE_rentable_area)
    m_AE_shape = PropBag.ReadProperty("AE_shape", m_def_AE_shape)
    m_AE_volume = PropBag.ReadProperty("AE_volume", m_def_AE_volume)
    m_AF_function = PropBag.ReadProperty("AF_function", m_def_AF_function)
    m_AT_hearing = PropBag.ReadProperty("AT_hearing", m_def_AT_hearing)
    m_AT_internal_sensitivity = PropBag.ReadProperty("AT_internal_sensitivity",
m_def_AT_internal_sensitivity)
    m_AT_recognition = PropBag.ReadProperty("AT_recognition", m_def_AT_recognition)
    m_AT_sight = PropBag.ReadProperty("AT_sight", m_def_AT_sight)
    m_AT_smell = PropBag.ReadProperty("AT_smell", m_def_AT_smell)
    m_AT_taste = PropBag.ReadProperty("AT_taste", m_def_AT_taste)
    txtDescription.Text = PropBag.ReadProperty("AA_description", "")
    m_AE_wall_space_ratio = PropBag.ReadProperty("AE_wall_space_ratio",
m_def_AE_wall_space_ratio)

End Sub

'Write property values to storage
Private Sub UserControl_WriteProperties(PropBag As PropertyBag)

    Call PropBag.WriteProperty("BackColor", m_BackColor, m_def_BackColor)
    Call PropBag.WriteProperty("ForeColor", m_ForeColor, m_def_ForeColor)
    Call PropBag.WriteProperty("Enabled", m_Enabled, m_def_Enabled)
    Call PropBag.WriteProperty("Font", m_Font, Ambient.Font)
    Call PropBag.WriteProperty("BackStyle", m_BackStyle, m_def_BackStyle)
    Call PropBag.WriteProperty("BorderStyle", m_BorderStyle, m_def_BorderStyle)
    Call PropBag.WriteProperty("AA_xdim", m_AA_xdim, m_def_AA_xdim)
    Call PropBag.WriteProperty("AA_ydim", m_AA_ydim, m_def_AA_ydim)
    Call PropBag.WriteProperty("AA_zdim", m_AA_zdim, m_def_AA_zdim)
    Call PropBag.WriteProperty("AA_scale", m_AA_scale, m_def_AA_scale)
    Call PropBag.WriteProperty("AA_unit", m_AA_unit, m_def_AA_unit)
    Call PropBag.WriteProperty("AA_name", lblDescription.Caption, "A")
    Call PropBag.WriteProperty("AE_construction_area", m_AE_construction_area,
m_def_AE_construction_area)
    Call PropBag.WriteProperty("AE_cost", m_AE_cost, m_def_AE_cost)
    Call PropBag.WriteProperty("AE_durability", m_AE_durability, m_def_AE_durability)
    Call PropBag.WriteProperty("AE_energy_use", m_AE_energy_use, m_def_AE_energy_use)
    Call PropBag.WriteProperty("AE_grossarea", m_AE_grossarea, m_def_AE_grossarea)
    Call PropBag.WriteProperty("AE_nettarea", m_AE_nettarea, m_def_AE_nettarea)
    Call PropBag.WriteProperty("AE_rentable_area", m_AE_rentable_area,
m_def_AE_rentable_area)
    Call PropBag.WriteProperty("AE_shape", m_AE_shape, m_def_AE_shape)
    Call PropBag.WriteProperty("AE_volume", m_AE_volume, m_def_AE_volume)
    Call PropBag.WriteProperty("AF_function", m_AF_function, m_def_AF_function)
    Call PropBag.WriteProperty("AT_hearing", m_AT_hearing, m_def_AT_hearing)
    Call PropBag.WriteProperty("AT_internal_sensitivity", m_AT_internal_sensitivity,
m_def_AT_internal_sensitivity)
    Call PropBag.WriteProperty("AT_recognition", m_AT_recognition,
m_def_AT_recognition)
    Call PropBag.WriteProperty("AT_sight", m_AT_sight, m_def_AT_sight)
    Call PropBag.WriteProperty("AT_smell", m_AT_smell, m_def_AT_smell)
    Call PropBag.WriteProperty("AT_taste", m_AT_taste, m_def_AT_taste)
    Call PropBag.WriteProperty("AA_description", txtDescription.Text, "")
    Call PropBag.WriteProperty("AE_wall_space_ratio", m_AE_wall_space_ratio,
m_def_AE_wall_space_ratio)

End Sub

Private Sub xSlide_Change()

    Dim Control_x As Double        ' Actual current total control width
    Dim Isometric_y As Double
    Dim Isometric_x As Double

    Isometric_y = AA - Ydim * 0.5
    Isometric_x = AA_ydim * 0.866

    ' The control is in 2D mode
    If (cmd2D_3D.Caption = "2") Then

        Control_x = xSlide.Value
        UserControl.Width = Control_x
        shpDesign.Width = Control_x
        lblDescription.Width = Control_x - 680.72
        cmdReduce.Left = Control_x - 427.71
        lblTo.Left = Control_x - 427.71
        lblCurrent.Left = (Control_x / 2#) - 190.771
        txtDescription.Width = Control_x - 445.783
        yzSlide.Left = Control_x - 149.498
        xSlide.Width = Control_x - 149.498

    'The control is in 3D mode
    Else

        Control_x = xSlide.Value
        UserControl.Width = Control_x + Isometric_x
        shpDesign.Width = Control_x

        lblDescription.Width = Control_x - 680.72
        cmdReduce.Left = Control_x - 427.71
        lblTo.Left = Control_x - 427.71
        lblCurrent.Left = (Control_x / 2#) - 190.771
        txtDescription.Width = Control_x - 445.783
        yzSlide.Left = Control_x - 149.498
        xSlide.Width = Control_x - 149.498

        linLine0_1.X2 = UserControl.Width
        linLine30_2.X1 = Control_x
        linLine30_2.X2 = UserControl.Width
        linLine90_1.X1 = UserControl.Width
        linLine90_1.X2 = UserControl.Width
        linLine30_3.X1 = Control_x
        linLine30_3.X2 = UserControl.Width

    End If

    AA_xdim = xSlide.Value

End Sub

Private Sub yzSlide_Change()

    Dim Control_y As Double         ' Actual current total control width
    Dim Control_z As Double         ' Actual current total control height
    Dim Isometric_y As Double
    Dim Isometric_x As Double

    Isometric_y = AA_ydim * 0.5
    Isometric_x = AA_ydim * 0.866

    ' The control is in 2D mode
    If (cmd2D_3D.Caption = "2") Then

        Control_y = yzSlide.Value
        UserControl.Height = Control_y
        shpDesign.Height = Control_y

        txtDescription.Height = Control_y - 680.72
        lblFrom.Top = Control_y - 325.301
        lblCurrent.Top = Control_y - 325.301
        lblTo.Top = Control_y - 325.301
        yzSlide.Height = Control_y - 149.498
        xSlide.Top = Control_y - 149.498

        AA_ydim = yzSlide.Value

    ' The control is in 3D mode
    Else
        Control_z = yzSlide.Value
        UserControl.Height = Control_z + Isometric_y
        shpDesign.Height = Control_z

        txtDescription.Height = Control_z - 680.72
        lblFrom.Top = (Control_z + Isometric_y) - 325.301
        lblCurrent.Top = (Control_z + Isometric_y) - 325.301
        lblTo.Top = (Control_z + Isometric_y) - 325.301
        yzSlide.Height = Control_z - 149.498
        xSlide.Top = (Control_z + Isometric_y) - 149.498

        linLine90_1.Y2 = Control_z
        linLine30_3.Y1 = UserControl.Height
        linLine30_3.Y2 = Control_z

        AA_zdim = yzSlide.Value

    End If

End Sub
```