```
Option Explicit

Private Const MODULE_NAME = "modShared"

Public Const U = "^"
Public Const vbCrLf2 = vbCrLf & vbCrLf
Public Const ADD = 0
Public Const EDIT = 1
Public Const DELETE = 2

' Framework components
Private iObjectsInitCount As Integer
Public oSession As CIA_CSS.CSS_Session
Public oEncounter As CSS_Encounter.Encounter
Public oPatient As CSS_Patient.Patient
Public oUser As CSS_User.User
Public oPrinter As vcPrint.vcPrintX
Public oSite As CSS_Site.Site
Public oVIM As CIA_VIM.VIM

' Security
Public g_bProvider As Boolean   ' PROVIDER key
Public g_bCAC As Boolean        ' BGOZ CAC key
Public g_bViewOnly As Boolean   ' BGOZ VIEW ONLY key
Public g_bEnabled As Boolean    ' Function-based key
Public g_sReason As String      ' Reason for restriction
Public g_bUsrNoEdit             ' Edit disabled for user
Public g_sClsNoEdit             ' Edit disabled for class

' External references
Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hwnd As Long, ByVal wMsg As Long
, ByVal wParam As Long, lparam As Any) As Long
Declare Function PostMessage Lib "user32" Alias "PostMessageA" (ByVal hwnd As Long, ByVal wMsg As Long
, ByVal wParam As Long, lparam As Any) As Long
Declare Function GetCursorPos Lib "user32" (pt As POINTAPI) As Long
Declare Function ScreenToClient Lib "user32" (ByVal hwnd As Long, pt As POINTAPI) As Long
Declare Function SetCapture Lib "user32" (ByVal hwnd As Long) As Long
Declare Function ReleaseCapture Lib "user32" () As Long
Declare Function WindowFromPoint Lib "user32" (pt As POINTAPI) As Long

Public Const EM_SCROLLCARET = &HB7&
Public Const EM_SETSEL = &HB1&
Public Const EM_UNDO = &HC7&
Public Const EM_REDO = &H454&
Public Const EM_REPLACESEL = &HC2&
Public Const LB_ITEMFROMPOINT = &H1A9&
Public Const WM_LBUTTONDOWN = &H201&
Public Const WM_LBUTTONUP = &H202&
Public Const WM_COMMAND = &H111&
Public Const WM_CUT = &H300&
Public Const WM_COPY = &H301&
Public Const WM_PASTE = &H302&

' Grid state info
Public Type TGridState
  sRowID As String
  iSortCol As Long
  iSortOrder As Integer
End Type

' Point structure
Public Type POINTAPI
  X As Long
  Y As Long
End Type

Public Sub ProcessShortcut(KeyCode As Integer, Shift As Integer, Handle As Long)
  If Shift = 2 Then
    Select Case KeyCode
      Case vbKeyA
        SendMessage Handle, EM_SETSEL, 0, -1

      Case vbKeyC
```

```
        SendMessage Handle, WM_COPY, 0, 0

      Case vbKeyV
        SendMessage Handle, WM_PASTE, 0, 0

      Case vbKeyX
        SendMessage Handle, WM_CUT, 0, 0

      Case vbKeyY
        SendMessage Handle, EM_REDO, 0, 0

      Case vbKeyZ
        SendMessage Handle, EM_UNDO, 0, 0

      Case Else
        Exit Sub
    End Select

    KeyCode = 0
  End If
End Sub

Public Sub FireClickEvent()
' This code does not yet work.
  Dim pt As POINTAPI, lparam As Long, hwnd As Long

  GetCursorPos pt
  hwnd = WindowFromPoint(pt)
  If hwnd = 0 Then Exit Sub
  ScreenToClient hwnd, pt
  lparam = pt.Y * 65536 + pt.X
  PostMessage hwnd, WM_LBUTTONDOWN, 1, lparam
  PostMessage hwnd, WM_LBUTTONUP, 1, lparam
End Sub

Public Function AppVersion() As String
  Dim s As String

  s = App.Major & "." & App.Minor & ".0." & App.Revision & "  " & App.Title
  If g_sReason <> "" Then s = s & vbCrLf2 & g_sReason
  AppVersion = s
End Function

Public Sub InitObjects(Optional LoadDummy As Boolean = True)
  On Error GoTo errHandle
  Dim i As Integer

  If LoadDummy Then
    Load frmDummy
    Unload frmDummy
    Set frmDummy = Nothing
  End If

  iObjectsInitCount = iObjectsInitCount + 1
  If iObjectsInitCount <> 1 Then Exit Sub
  Dim oServer As New CIA_CSS.CSS_Server
  Set oSession = oServer.Session
  Set oServer = Nothing
  Set oVIM = oSession.FindObjectByProgID("CIA_VIM.VIM", Nothing)

  If oVIM Is Nothing Then
    If oSession.HostPort = 0 Then
      oSession.Connect ""
    End If
  End If

  Set oPatient = oSession.FindServiceByProgID("CSS_Patient.Patient")
  Set oUser = oSession.FindServiceByProgID("CSS_User.User")
  Set oSite = oSession.FindServiceByProgID("CSS_Site.Site")
  Set oEncounter = oSession.FindServiceByProgID("CSS_Encounter.Encounter")
  Set oPrinter = oSession.FindServiceByProgID("vcPrint.vcPrintX")

  If oVIM Is Nothing Then
```

```
    If oPatient.Handle = 0 Then
      oPatient.Select
    End If

    If oEncounter.Location = 0 Then
      oEncounter.Prepare ofProvider
    End If
  End If

  Exit Sub

errHandle:
  ErrorTrap MODULE_NAME, "InitObjects"
End Sub

Public Sub InitSecurity(Optional sDisableParam As String, Optional sFunctionKey As String)
' Initialize Public variables used for access restriction.
' sDisableParameter is a parameter that restricts write access by user or by user class.
' sFunctionKey is a security key that restricts write access by specific function.

  Dim s As String, sKeys As String, sParams As String

  s = "PROVIDER|BGOZ CAC|BGOZ VIEW ONLY|" & sFunctionKey & U & sDisableParam
  s = CallRPCStr("BGOUTL CHKSEC", s)
  sKeys = Piece(s, U)
  sParams = Piece(s, U, 2)
  g_bProvider = Piece(sKeys, "|") = "1"
  g_bCAC = Piece(sKeys, "|", 2) = "1"
  g_bViewOnly = Piece(sKeys, "|", 3) = "1"
  g_bEnabled = Piece(sKeys, "|", 4) = "1" Or sFunctionKey = ""
  g_bUsrNoEdit = Piece(sParams, "~") = "1"
  g_sClsNoEdit = Piece(sParams, "~", 2)

  If g_bViewOnly Then
    g_sReason = "You have the 'BGOZ VIEW ONLY' security key"
  ElseIf Not g_bProvider And Not g_bEnabled Then
    If sFunctionKey = "" Then
      g_sReason = "You do not have the 'PROVIDER' key"
    Else
      g_sReason = "You do not have either the 'PROVIDER' or the '" & sFunctionKey & "' security keys"
    End If
  ElseIf g_bUsrNoEdit Then
    g_sReason = "You have been assigned the '" & sDisableParam & "' parameter"
  ElseIf g_sClsNoEdit <> "" Then
    g_sReason = "You are a member of the user class '" & g_sClsNoEdit & "' which has been assigned the
 '" & sDisableParam & "' parameter"
  Else
    g_sReason = ""
    Exit Sub
  End If

  g_sReason = g_sReason & ", thus you cannot make modifications."
End Sub

Public Function CheckSecurity(Optional Silent As Boolean) As Boolean
  If g_sReason = "" Then
    CheckSecurity = True
  Else
    CheckSecurity = False
    If Not Silent Then ShowError g_sReason, TX_ERR_PERMISSIONS
  End If
End Function

Public Sub ReleaseObjects()
  On Error Resume Next
  If iObjectsInitCount > 0 Then iObjectsInitCount = iObjectsInitCount - 1
  If iObjectsInitCount > 0 Then Exit Sub
  Set oSession = Nothing
  Set oPatient = Nothing
  Set oUser = Nothing
  Set oSite = Nothing
  Set oEncounter = Nothing
  Set oPrinter = Nothing
```

```
  Set oVIM = Nothing
End Sub

Public Function Confirm(Text As String, Optional Caption As String) As Boolean
  If Caption = "" Then Caption = "Confirm"
  Confirm = MsgBox(Text, vbQuestion + vbYesNo + vbDefaultButton2, Caption) = vbYes
End Function

Public Sub Status(Optional Text As String)
  If oSession Is Nothing Then Exit Sub
  oSession.EventFireLocal "STATUS", Text
End Sub

Private Sub ShowMessage(Text As String, Caption As String, Severity As Integer)
  If Caption = "" Then
    Select Case Severity
      Case vbCritical
        Caption = "Error"

      Case vbInformation
        Caption = "Information"

      Case vbExclamation
        Caption = "Warning"
    End Select
  End If

  If Text <> "" Then MsgBox Text, Severity, Caption
End Sub

Public Sub ShowError(Text As String, Optional Caption As String)
  ShowMessage Text, Caption, vbCritical
End Sub

Public Sub ShowInfo(Text As String, Optional Caption As String)
  ShowMessage Text, Caption, vbInformation
End Sub

Public Sub ShowWarning(Text As String, Optional Caption As String)
  ShowMessage Text, Caption, vbExclamation
End Sub

Public Function ErrorCheck(Value As String, Optional Caption As String) As Boolean
  If ErrorCode(Value) <> 0 Then
    ShowError Piece(Value, U, 2), Caption
    Status
    ErrorCheck = True
  Else
    ErrorCheck = False
  End If
End Function

Public Function ErrorCode(Value As String) As Long
  Dim i As Long
  On Error Resume Next
  i = Val(Piece(Value, U))
  ErrorCode = IIf(i < 0, -i, 0)
End Function

Public Sub ErrorTrap(Module As String, Method As String)
  Screen.MousePointer = vbDefault
  Status
  ShowError "Error in module " & Module & " at " & Method & vbCrLf2 & Err.Description, "System Error"
End Sub

Public Function CallRPCStr(sRpc As String, Optional Param As Variant = "") As String
  On Error GoTo errHandle
  Screen.MousePointer = vbHourglass
  CallRPCStr = oSession.CallRPCStr(sRpc, Param)
  Screen.MousePointer = vbDefault
  Exit Function

errHandle:
```

```
  Screen.MousePointer = vbDefault
  ShowError "Error in remote procedure call: " & sRpc & vbCrLf & Err.Description, TC_ERR_SYSTEM
End Function

Public Function CallRPCList(sRpc As String, Optional Param As Variant = "") As String
  On Error GoTo errHandle
  Screen.MousePointer = vbHourglass
  CallRPCList = oSession.CallRPCText(sRpc, Param)
  Screen.MousePointer = vbDefault
  Exit Function

errHandle:
  Screen.MousePointer = vbDefault
  ShowError "Error in remote procedure call: " & sRpc & vbCrLf & Err.Description, TC_ERR_SYSTEM
End Function

Public Sub ShowReport(Text As String, Caption As String)
  On Error GoTo errHandle
  oPrinter.Preview Text, Caption, "", "", "", 80, True, True
  Exit Sub

errHandle:
  ShowError "Error displaying requested report: " & Caption & vbCrLf & Err.Description, TC_ERR_SYSTEM
End Sub

Public Sub VisitDetail(ByVal VisitIEN As String)
  On Error GoTo errHandle
  Dim sRpc As String

  If Trim(VisitIEN) = "" Then Exit Sub
  sRpc = CallRPCList("BGOUTL GETRPT", VisitIEN)
  ShowReport sRpc, "Visit Detail"
  Exit Sub

errHandle:
  ErrorTrap MODULE_NAME, "VisitDetail"
End Sub

Public Function FindComboBoxItem(AControl As ComboBox, Text As String) As Integer
  Dim i As Integer

  With AControl
    .ListIndex = -1

    For i = 0 To .ListCount - 1
      If SameText(Text, .List(i)) Then
        .ListIndex = i
        Exit For
      End If
    Next i

    FindComboBoxItem = .ListIndex
  End With

End Function

Public Function FindComboBoxData(AControl As ComboBox, Data As String) As Integer
  On Error GoTo errHandle
  Dim i As Integer, Value As Long

  With AControl
    .ListIndex = -1
    Value = CLng(Data)

    For i = 0 To .ListCount - 1
      If .ItemData(i) = Value Then
        .ListIndex = i
        Exit For
      End If
    Next i

    FindComboBoxData = .ListIndex
  End With
```

```
  Exit Function

errHandle:
  FindComboBoxData = -1
End Function

Public Function SplitEx(TextString As String, Delimiter As String, Optional MinSize As Integer) As Str
ing()
' Same as Split, but insures a minimum array size upon return
  Dim StrArray() As String
  StrArray = Split(TextString, Delimiter)
  If UBound(StrArray) < MinSize Then ReDim Preserve StrArray(MinSize)
  SplitEx = StrArray
End Function

Public Function PieceCount(TextString As String, Delimiter As String) As Integer
  Dim i As Integer, Count As Integer

  Count = 0

  If TextString <> "" And Delimiter <> "" Then
    i = 1

    Do
      Count = Count + 1
      i = InStr(i, TextString, Delimiter, vbBinaryCompare)
      If i > 0 Then i = i + Len(Delimiter)
    Loop While i > 0
  End If

  PieceCount = Count
End Function

Public Function Piece(ByVal TextString As String, ByVal Delimiter As String, Optional ByVal PieceNumbe
r As Integer = 1)
'Returns the nth Delimiter-delimited piece of TextString
'Where n=PieceNumber
'Corresponds to Mumps $Piece function

  Piece = ""

  If PieceNumber = 0 Or TextString = Delimiter Then Exit Function

  If InStr(1, TextString, Delimiter) = 0 Then
    If PieceNumber = 1 Then Piece = TextString
    Exit Function
  End If

  If PieceNumber = 1 Then
    Piece = Left$(TextString, InStr(1, TextString, Delimiter) - 1)
    Exit Function
  End If

  Do While PieceNumber > 1
    If InStr(1, TextString, Delimiter) = 0 Then Exit Function
    TextString = Right$(TextString, Len(TextString) - InStr(1, TextString, Delimiter) - Len(Delimiter)
 + 1)
    PieceNumber = PieceNumber - 1
  Loop

  Piece = Piece(TextString, Delimiter, 1)
End Function

Public Sub SetPiece(ByRef TextString As String, ByVal Delimiter As String, ByVal PieceNumber As Intege
r, ByVal Value As String)
  Dim i As Integer, j As Integer, s As String

  i = PieceCount(TextString, Delimiter)

  If PieceNumber > i Then TextString = TextString & String(PieceNumber - i, Delimiter)
  i = PieceCount(TextString, Delimiter)
  s = ""
```

```
  For j = 1 To i
    s = s & IIf(j = 1, "", Delimiter) & IIf(j = PieceNumber, Value, Piece(TextString, Delimiter, j))
  Next j

  TextString = s
End Sub

Public Function Xlate(ByVal Value As String, FromStr As String, Optional ToStr As String) As String
  Dim i As Integer

  For i = 1 To Len(FromStr)
    Value = Replace(Value, Mid$(FromStr, i, 1), Mid$(ToStr, i, 1))
  Next i

  Xlate = Value
End Function

'============================================================
' This function creates an RPC parameter list (variant array)
' from any number of arguments or data types passed to it.
'
' For example, the following code creates an RPC parameter
' list from 5 totally different data types, then calls' CallRPCList.
'   Dim params as Variant
'   params = CreateRPCParamList("abc", 1, today, 4/5, x)
'   result = oSession.CallRPCText("Some RPC Name", params)
'============================================================
Public Function CreateRPCParamList(ParamArray params() As Variant) As Variant
  CreateRPCParamList = params
End Function

Public Sub SetParam(Param As String, Value As Variant)
' Retrieve BGO settings parameter.  These are cached once retrieved.
  Dim s As String

  s = CallRPCStr("BGOUTL SETPARM", Param & U & Value)

  If Not ErrorCheck(s, "Setting Parameter " & Param) Then
    oSession.Param("BGOParam_" & Xlate(Param, "-", "_")) = Value
  End If
End Sub

Public Function GetParam(Param As String, Optional Default As String) As String
  Dim s As Variant, v As String

  v = "BGOParam_" & Xlate(Param, "-", "_")
  s = oSession.Param(v)

  If IsNull(s) Then
    s = CallRPCStr("BGOUTL GETPARM", Param)
    oSession.Param(v) = s
  End If

  GetParam = IIf(s = "", Default, s)
End Function

Public Function GetSysParam(Param As String, Optional Default As String, Optional ByVal Instance As St
ring) As String
  Dim s As Variant, v As String

  If Instance = "" Then Instance = "1"
  v = "CIAParam_" & Xlate(Param, "- ", "__") & "_" & Instance
  s = oSession.Param(v)

  If IsNull(s) Then
    s = CallRPCStr("CIAVMRPC GETPAR", CreateRPCParamList(Param, "", Instance))
    oSession.Param(v) = s
  End If

  GetSysParam = IIf(s = "", Default, s)
End Function

Public Function SetSysParam(Param As String, Value As String) As Boolean
```

```
  Dim s As String

  s = Piece(CallRPCStr("CIAVMRPC SETPAR", CreateRPCParamList(Param, Value, "USR")), U, 2)

  If s <> "" Then ShowError s, "Error Saving Parameter"
  SetSysParam = s = ""
End Function

Public Function EndsText(ByVal Text As String, ByVal Substring As String, Optional CaseSensitive As Bo
olean) As Boolean
  EndsText = StrComp(Right(Text, Len(Substring)), Substring, IIf(CaseSensitive, vbBinaryCompare, vbTex
tCompare)) = 0
End Function

Public Function StartsText(ByVal Text As String, ByVal Substring As String, Optional CaseSensitive As
Boolean) As Boolean
  StartsText = StrComp(Left(Text, Len(Substring)), Substring, IIf(CaseSensitive, vbBinaryCompare, vbTe
xtCompare)) = 0
End Function

Public Function SameText(Text1 As String, Text2 As String) As Boolean
  SameText = StrComp(Text1, Text2, vbTextCompare) = 0
End Function

Public Function ContainsText(Text As String, Substring As String) As Boolean
  ContainsText = InStr(1, Text, Substring, vbTextCompare) > 0
End Function

Public Function StrToIntDef(Value As String, Optional Default As Long) As Long
  On Error GoTo errHandle
  StrToIntDef = CLng(Value)
  Exit Function

errHandle:
  StrToIntDef = Default
End Function

Public Function IntComp(Value1 As Long, Value2 As Long) As Integer
  If Value1 = Value2 Then
    IntComp = 0
  ElseIf Value1 > Value2 Then
    IntComp = 1
  Else
    IntComp = -1
  End If
End Function

Public Function DateComp(Value1 As String, Value2 As String) As Integer
  Dim Date1 As Date, Date2 As Date
  Dim IsDate1 As Boolean, IsDate2 As Boolean

  IsDate1 = IsDate(Value1)
  IsDate2 = IsDate(Value2)
  If IsDate1 Then Date1 = CDate(Value1)
  If IsDate2 Then Date2 = CDate(Value2)

  If Not IsDate1 And Not IsDate2 Then
    DateComp = StrComp(Value1, Value2, vbTextCompare)
  ElseIf Not IsDate1 Then
    DateComp = -1
  ElseIf Not IsDate2 Then
    DateComp = 1
  ElseIf Date1 > Date2 Then
    DateComp = 1
  ElseIf Date1 < Date2 Then
    DateComp = -1
  Else
    DateComp = 0
  End If
End Function

Public Function SetProperCase(ByVal s As String) As String
  s = Replace(s, ",", ", ")
```

```
  s = Replace(s, "-", "- ")
  s = Replace(s, ";", "; ")
  s = Replace(s, ":", ": ")
  s = Replace(s, ".", ". ")
  s = StrConv(s, vbProperCase)
  s = Replace(s, ", ", ",")
  s = Replace(s, "- ", "-")
  s = Replace(s, "; ", ";")
  s = Replace(s, ": ", ":")
  s = Replace(s, ". ", ".")
  SetProperCase = s
End Function

Public Function DateToFMDate(ByVal sDate As String, Optional IncludeTime As Boolean) As String
' Converts a date from mm/dd/yyyy format to FM format
  On Error GoTo errHandle
  Dim s As String, m As Double, d As Double, yr As Double, dt As Double, dat As Date
  DateToFMDate = ""
  If sDate = "" Then Exit Function
  dat = CDate(sDate)
  s = Format(dat, "mm/dd/yyyy")
  m = Val(Piece(s, "/", 1))
  d = Val(Piece(s, "/", 2))
  yr = Val(Piece(s, "/", 3)) - 1700

  If d > 0 And d < 32 And m > 0 And m < 13 And yr > 0 Then
    dt = yr * 10000 + m * 100 + d

    If IncludeTime Then
      s = Format(dat, "hhnnss")
      dt = dt + Val(s) / 1000000
    End If

    DateToFMDate = CStr(dt)
  End If
  Exit Function

errHandle:
  ErrorTrap MODULE_NAME, "DateToFMDate"
End Function

Public Function NormalizeDate(ByVal sDate As String, Optional IncludeTime As Boolean) As String
  Dim d As Date

  If sDate = "" Then
    NormalizeDate = ""
    Exit Function
  ElseIf IsNumeric(sDate) Then  ' FileMan format
    sDate = CStr((CDbl(sDate) + 17000000) * 1000000)
    d = DateSerial(Left(sDate, 4), Mid(sDate, 5, 2), Mid(sDate, 7, 2)) _
      + TimeSerial(Mid(sDate, 9, 2), Mid(sDate, 11, 2), Mid(sDate, 13, 2))
  Else
    sDate = Xlate(sDate, "@", " ")

    If IsDate(sDate) Then
      d = CDate(sDate)
    Else
      NormalizeDate = ""
      Exit Function
    End If
  End If

  If IncludeTime Then
    NormalizeDate = Format(d, "mm/dd/yyyy hh:mm")
  Else
    NormalizeDate = Format(d, "mm/dd/yyyy")
  End If
End Function

Public Function ListItemFromPoint(hwnd As Long, X As Single, Y As Single) As Integer
  Dim lparam As Long, lResult As Long

  lparam = (CInt(Y / Screen.TwipsPerPixelY) * 2 ^ 16) + CInt(X / Screen.TwipsPerPixelX)
```

```
  lResult = SendMessage(hwnd, LB_ITEMFROMPOINT, 0, ByVal lparam)

  If (lResult \ 2 ^ 16) <> 0 Then
    ListItemFromPoint = -1
  Else
    ListItemFromPoint = CInt(lResult)
  End If

End Function

Public Sub UpdateTooltip(List As ListBox, X As Single, Y As Single)
  Dim i As Integer

  i = ListItemFromPoint(List.hwnd, X, Y)

  If i >= 0 Then
    List.ToolTipText = Replace(List.List(i), vbTab, " - ")
  Else
    List.ToolTipText = ""
  End If
End Sub

Public Function EnforceRange(Value As Integer, Lower As Integer, Higher As Integer) As Integer
  If Value < Lower Then
    EnforceRange = Lower
  ElseIf Value > Higher Then
    EnforceRange = Higher
  Else
    EnforceRange = Value
  End If
End Function
```