```
ThisWorkbook - 1

'.....................................................................................
'.....................................................................................
'.......        Developer:              William H. White (consultant)
'.......        With:                   TEKsystems Inc.
'.......                                www.teksystems.com
'.......        For:                    AIG
'.......                                Financial Information Systems
'.......                                1 NY Plaza, 15th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         8/8/2013
'.......        Proj finished:          11/21/2013
'.......        Proj revised:           1/15/2014
'.....................................................................................
'.....................................................................................

'.....................................................................................
'.........................  Class-level COMMENTS  ....................................
'Commentary...
'.....................................................................................
'.....................................................................................


'*************************************************************************************
'*************************************************************************************
'*************************  CLASS-LEVEL DECLARATIONS  ************************************
'*************************************************************************************
'*************************************************************************************
Option Explicit
Option Compare Binary '...use cAsE senSITive string comparisons




'*************************************************************************************
'*************************************************************************************
'*******************************  PRIVATE MEMBERS  ***********************************
'*************************************************************************************
'*************************************************************************************

'**************************************
'MODULE/CLASS-WIDE CONSTANTS
'**************************************
'Strings...
Private Const m_sMODULE_NAME As String = "ThisWorkbook"




'=============================================================
'======================  EVENTS  ============================
'=============================================================
'=============================================================================
Private Sub Workbook_Open()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "Workbook_Open()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim vsbValWshs As XlSheetVisibility

'Allow user to halt macro execution by hitting Esc or Ctrl+Break.  IF a user _
does this VBA will treat the action as a run-time error, with _
Err.Number = 18 (which is the value we have set for g_lUSER_CANCEL).  A user _
cancel will now be routed through our central error handler (which, in turn, _
will ignore the error)...
```

ThisWorkbook - 2

```
Application.EnableCancelKey = xlErrorHandler

On Error GoTo ErrorHandler

'Maximize workbook window...
Windows(ThisWorkbook.name).WindowState = xlMaximized
WshGoodData.Activate

'Instantiate some globals variables (done as follows for those instances _
in which we are re-running Workbook_Open() whilst debugging)...
g_bDebugMode = False
g_bStepThruMode = False
If g_cMsgBoxHandler Is Nothing Then Set g_cMsgBoxHandler = New MsgBoxHandler

g_bMandDimensOnly = _
        ThisWorkbook.CustomDocumentProperties("HandleOnlyMandDimens").value

If Not ThisWorkbook.CustomDocumentProperties("Production Mode").value Then
    'Prompt for running in debug mode...
    Dim ansRunInDebugMode As VbMsgBoxResult
    g_cMsgBoxHandler.vSetRunDebugModeMsg
    ansRunInDebugMode = g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbYesNo)
    If ansRunInDebugMode = vbYes Then g_bDebugMode = True
End If

If g_bDebugMode Then
    'Prompt for stepping through code...
    Dim ansStepThruMode As VbMsgBoxResult
    g_cMsgBoxHandler.vSetStepThruModeMsg
    ansStepThruMode = g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbYesNo)
    If ansStepThruMode = vbYes Then g_bStepThruMode = True
End If

'Set val wsh visibility...
If g_bDebugMode Then
    vsbValWshs = xlSheetVisible
Else
    vsbValWshs = xlSheetVeryHidden
End If
WshSLDimLocs.Visible = vsbValWshs
WshSLValLists.Visible = vsbValWshs
WshValLists.Visible = vsbValWshs

'Instantiate some objects....
Set g_cObjFactory = New ObjectFactory

'Set a default...
RibbonHandler.SetNextStage projNxtStgOpenSrc

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
            ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'================================================================================
Private Sub Workbook_BeforeSave(ByVal SaveAsUI As Boolean, Cancel As Boolean)
```

```
'*** Because the signature of this method is a given, for our error-handling _
purposes we have no choice to treat this as an entry-point procedure...

'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "Workbook_BeforSave()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...


On Error GoTo ErrorHandler

'Call a lower-level sub-routine.  Make sure the SUB has a ByRef boolean _
parameter for error-handling...
'''NonEntryPointProcedure RanOkRESULT:= bCalledProcedureRanOk
'''If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
            ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'================================================================================
Private Sub Workbook_BeforeClose(Cancel As Boolean)

'Generally, we do NOT want to do any error-handling here, nor do we _
want remaining code execution here to be interrupted, so...
On Error Resume Next

'Clean-up code goes here...

End Sub
'================================================================================
'================================================================================
'================================================================================
'================================================================================
```

```
frmAddRemWshTypes - 1

'.....................................................................................
'.....................................................................................
'.......        Developer:              William H. White (consultant)
'.......        With:                   TEKsystems Inc.
'.......                                www.teksystems.com
'.......        For:                    AIG
'.......                                Financial Information Systems
'.......                                1 NY Plaza, 15th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         8/28/2013
'.......        Proj finished:          11/21/2013
'.......        Proj revised:           1/15/2014
'.....................................................................................
'.....................................................................................

'.....................................................................................
'.........................  Form-level COMMENTS  .....................................
'Commentary...
'.....................................................................................
'.....................................................................................



'*************************************************************************************
'*************************************************************************************
'*************************  FORM-LEVEL DECLARATIONS  *************************************
'*************************************************************************************
'*************************************************************************************
Option Explicit
Option Compare Binary '...use cAsE senSITive string comparisons



'*************************************************************************************
'*************************************************************************************
'*******************************  PRIVATE MEMBERS  ***********************************
'*************************************************************************************
'*************************************************************************************

'***************************************
'FORM-WIDE CONSTANTS
'***************************************
'Strings...
Private Const m_sFORM_NAME As String = "frmAddRemWshTypes"


'***************************************
'MODULE/Form-WIDE VARIABLES
'***************************************
'Booleans...
Private m_bUserCancel As Boolean
Private m_bALLTypeSpecified As Boolean




'=========================================================
'==================  (Private) EVENTS  ===================
'=========================================================
'=====================================================================================
Private Sub UserForm_Initialize()
'[VBA QUIRK:  Errors in Intialize, Activate, and Terminate event handlers _
cannot be handled by procedures which create or destroy instances of the _
class. If any such error-handling IS required, create a separate public _
```

```
procedure - e.g., the Initialize procedure, below.]

End Sub
'============================================================================
Private Sub UserForm_Terminate()
'[VBA QUIRK:  Errors in Intialize, Activate, and Terminate event handlers _
cannot be handled by procedures which create or destroy instances of the _
class. If any such error-handling IS required, create a separate public _
procedure - e.g., the Initialize procedure, below.]

'Since there's really nothing useful an error-handler can do at this point...
On Error Resume Next

End Sub
'============================================================================
Private Sub btnAddType_Click()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "btnAddType_Click()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

On Error GoTo ErrorHandler

'If g_bStepThruMode Then Stop
If StrComp(Trim(txtType), vbNullString) = 0 Then GoTo ExitPoint

g_cFormsHandler.UserAddsWshType RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'============================================================================
Private Sub btnRemoveType_Click()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "btnRemoveType_Click()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim bLeaveFormOpen As Boolean

On Error GoTo ErrorHandler
If g_bStepThruMode Then Stop
g_cFormsHandler.UserRemovesWshTypes RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    Dim bRetraceErrorMode As Boolean
```

```
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If

End Sub
'==========================================================================
Private Sub btnOK_Click()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "btnOK_Click()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

m_bALLTypeSpecified = False '...reset, just in case

g_cFormsHandler.UserOKClosesAddRemWshTypesFormMaestro _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
m_bUserCancel = False

Exit Sub

ErrorHandler:
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'==========================================================================
Private Sub btnCancel_Click()
    'NOTE 1:  MSDN has some notes on creating and setting the following _
private boolean.  I still have no idea about how to work with it...
'm_bUserCancel = True
    'NOTE 2:  We can't use our centralized error-trapping with a call to _
g_cFormsHandler.UserWantsToCxlCloseForm m_sFORM_NAME.  That procedure may _
wind up Unload-ing this form.  If so, there would be no object to hold our _
typical bCalledProcedureRanOK boolean.  As such, that value would flip to _
its default of FALSE.  Remaining code in this btnCancel_Click() WOULD _
continue to run.  However, with the boolean flipping to FALSE we would _
always generate an error.  Therefore, as noted in the _
g_cFormsHandler.UserWantsToCxlCloseForm m_sFORM_NAME sub, we effectively _
treat THAT sub as our entry point...
If g_bStepThruMode Then Stop
g_cFormsHandler.UserWantsToCxlCloseForm m_sFORM_NAME
End Sub
'==========================================================================
Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
'.....................
'MSDN on...
```

```
'Cancel:  "Setting this argument to any value other than 0 [i.e., other than _
FALSE, i.e., "Setting this argument to TRUE"...] stops the QueryClose event in _
all loaded user forms and prevents the UserForm and application from closing."
'CloseMode:  "A value or constant indicating the cause of the QueryClose event"
'.....................

'NOTE:  vbFormControlMenu ==> _
    The user has chosen the Close command from the Control menu on the UserForm.
'NOTE:  False === 0

'Route any X-close button calls through the btnCancel_Click procedure...
If CloseMode = vbFormControlMenu Then
    g_cMsgBoxHandler.SetPlsUseOKCxlBtnsMsg
    g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbExclamation
    Cancel = True 'i.e., -1
End If
End Sub




'*****************************************************************************
'*****************************************************************************
'*****************************  PUBLIC MEMBERS  ******************************
'*****************************************************************************
'*****************************************************************************

'============================================================
'======================  PROPERTIES  ========================
'============================================================
'=============================================================================
'Property UserCancel (read-only)...
Property Get UserCancel() As Boolean
    UserCancel = m_bUserCancel
End Property
'Property ALLTypeSpecified...
Property Get ALLTypeSpecified() As Boolean
    ALLTypeSpecified = m_bALLTypeSpecified
End Property
Property Let ALLTypeSpecified(value As Boolean)
    m_bALLTypeSpecified = value
End Property

'=============================================================================
'=============================================================================
'=============================================================================
'=============================================================================
'=============================================================================
'=============================================================================
```

```
'.....................................................................................
'.....................................................................................
'.......        Developer:              William H. White (consultant)
'.......        With:                   TEKsystems Inc.
'.......                                www.teksystems.com
'.......        For:                    AIG
'.......                                Financial Information Systems
'.......                                1 NY Plaza, 15th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         9/1/2013
'.......        Proj finished:          11/21/2013
'.......        Proj revised:           1/15/2014
'.....................................................................................
'.....................................................................................

'.....................................................................................
'.........................  Form-level COMMENTS  .....................................
'Commentary...
'.....................................................................................
'.....................................................................................



'*************************************************************************************
'*************************************************************************************
'*************************  FORM-LEVEL DECLARATIONS  *************************************
'*************************************************************************************
'*************************************************************************************
Option Explicit
Option Compare Binary '...use cAsE senSITive string comparisons



'*************************************************************************************
'*************************************************************************************
'*******************************  PRIVATE MEMBERS  ***********************************
'*************************************************************************************
'*************************************************************************************

'***************************************
'FORM-WIDE CONSTANTS
'***************************************
'Strings...
Private Const m_sFORM_NAME As String = "frmTypeWshAssoc"


'***************************************
'MODULE/Form-WIDE VARIABLES
'***************************************
'Worksheet Types...
Private m_collWshTypes As WshTypes

'Booleans...
Private m_bUserCancel As Boolean



'=========================================================
'==================  (Private) EVENTS  ===================
'=========================================================
'=====================================================================================
Private Sub UserForm_Initialize()
'[VBA QUIRK:  Errors in Intialize, Activate, and Terminate event handlers _
cannot be handled by procedures which create or destroy instances of the _
class. If any such error-handling IS required, create a separate public _
procedure - e.g., the Initialize procedure, below.]
```

```
frmTypeWshAssoc - 2


End Sub
'================================================================================
Private Sub UserForm_Terminate()
'[VBA QUIRK:  Errors in Intialize, Activate, and Terminate event handlers _
cannot be handled by procedures which create or destroy instances of the _
class. If any such error-handling IS required, create a separate public _
procedure - e.g., the Initialize procedure, below.]

'Since there's really nothing useful an error-handler can do at this point...
On Error Resume Next

End Sub
'================================================================================
Private Sub btnAssocWshWithType_Click()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "btnAssocWshWithType_Click()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

'If g_bStepThruMode Then Stop
g_cFormsHandler.AddRemoveWshsToWshTypeAssocAndUpdateListBoxes _
        AssocDisassoc:=projAssoc, _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'================================================================================
Private Sub btnRemove_Click()
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "btnRemove_Click()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

If g_bStepThruMode Then Stop
g_cFormsHandler.AddRemoveWshsToWshTypeAssocAndUpdateListBoxes _
        AssocDisassoc:=projDisassoc, _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
```

```
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'================================================================================
Private Sub btnCancel_Click()
    'NOTE 1:  MSDN has some notes on creating and setting the following _
private boolean.  I still have no idea about how to work with it...
'm_bUserCancel = True
    'NOTE 2:  We can't use our centralized error-trapping with a call to _
g_cFormsHandler.UserWantsToCxlCloseForm m_sFORM_NAME.  That procedure may _
wind up Unload-ing this form.  If so, there would be no object to hold our _
typical bCalledProcedureRanOK boolean.  As such, that value would flip to _
its default of FALSE.  Remaining code in this btnCancel_Click() WOULD _
continue to run.  However, with the boolean flipping to FALSE we would _
always generate an error.  Therefore, as noted in the _
g_cFormsHandler.UserWantsToCxlCloseForm m_sFORM_NAME sub, we effectively _
treat THAT sub as our entry point...
g_cFormsHandler.UserWantsToCxlCloseForm m_sFORM_NAME
End Sub
'================================================================================
Private Sub btnOK_Click()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "btnOK_Click()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim bLeaveFormOpen As Boolean


On Error GoTo ErrorHandler
If g_bStepThruMode Then Stop
g_cFormsHandler.PostWshTypeToWshAssocToSrcDataWbk _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
m_bUserCancel = False
Exit Sub

ErrorHandler:
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'================================================================================
Private Sub cmbWshTypes_Change()
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "btnRemove_Click()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

'If g_bStepThruMode Then Stop
g_cFormsHandler.UpdateAssociatedWshListBoxForSelectedWshType _
```

```
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If

End Sub
'============================================================================
Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
'......................
'MSDN on...
'Cancel:  "Setting this argument to any value other than 0 [i.e., other than _
FALSE, i.e., "Setting this argument to TRUE"...] stops the QueryClose event in _
all loaded user forms and prevents the UserForm and application from closing."
'CloseMode:  "A value or constant indicating the cause of the QueryClose event"
'......................

'NOTE:  vbFormControlMenu ==> _
    The user has chosen the Close command from the Control menu on the UserForm.
'NOTE:  False === 0

'Route any X-close button calls through the btnCancel_Click procedure...
If CloseMode = vbFormControlMenu Then
    g_cMsgBoxHandler.SetPlsUseOKCxlBtnsMsg
    g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbExclamation
    Cancel = True 'i.e., -1
End If
End Sub




'****************************************************************************
'****************************************************************************
'*****************************  PUBLIC MEMBERS  *****************************
'****************************************************************************
'****************************************************************************

'============================================================
'======================  PROPERTIES  ======================
'============================================================
'============================================================================
'Property UserCancel (read-only)...
Property Get UserCancel() As Boolean
    UserCancel = m_bUserCancel
End Property
'Property WorksheetTypes (read-only)...
Property Get WorksheetTypes() As WshTypes
    Set WorksheetTypes = m_collWshTypes
End Property
```

```
'=============================================================
'==================  (Public) PROCEDURES  ==================
'=============================================================
'=============================================================================
Public Sub CloneWshTypesColl(ByRef RanOkRESULT As Boolean)
'[NOTE:  Use this procedure whenever there is start-up code which might _
possibly fail.  Put such code here, which is a procedure which must be _
called expicitly, because, as noted above, errors which might occur in _
the Class_Initialize() method CANNOT be trapped by a calling procedure.]

'Error-handling declarations...
Const sSOURCE As String = "CloneWshTypesColl()"
Const bPROCEDURE_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until an error is encountered...
RanOkRESULT = True

Set m_collWshTypes = _
        g_cDimHandler.WorksheetTypes.ICloneable_Clone(bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
Exit Sub

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
            ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bPROCEDURE_IS_ENTRY_PT
    If bRetraceErrorMode Then
        Else
        Resume ExitPoint
    End If
End Sub
'=============================================================================
'=============================================================================
'=============================================================================
'=============================================================================
'=============================================================================
'=============================================================================
```

```
'.....................................................................................
'.......        Developer:              William H. White (consultant)
'.......        With:                   TEKsystems Inc.
'.......                                www.teksystems.com
'.......        For:                    AIG
'.......                                Financial Information Systems
'.......                                1 NY Plaza, 15th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         8/26/2013
'.......        Proj finished:          11/21/2013
'.......        Proj revised:           1/15/2014
'.....................................................................................
'.....................................................................................


'*************************************************************************************
'*************************************************************************************
'*************************  FORM-LEVEL DECLARATIONS  *************************
'*************************************************************************************
'*************************************************************************************
Option Explicit
Option Compare Binary '...use cAsE senSITive string comparisons




'*************************************************************************************
'*************************************************************************************
'*******************************  PRIVATE MEMBERS  *******************************
'*************************************************************************************
'*************************************************************************************

'***************************************
'FORM-WIDE CONSTANTS
'***************************************
'Strings...
Private Const m_sFORM_NAME As String = "frmWbkMandDimens"



'***************************************
'MODULE/Form-WIDE VARIABLES
'***************************************
'Dictionaries...
Private m_dictRefEditByTxtBox As CtrlNmByCtrlNm

'Collections...
Private m_collTextBoxesByCtrlNm As ByNmFormControls
Private m_collRefEditsByCtrlNm As ByNmFormControls
Private m_collTxtBoxByDimenNm As ByDimenNmFrmCtrls
Private m_collRefEditByDimNm As ByDimenNmFrmCtrls
Private m_collDimensPopulatedOnOpen As Dimensions


'Booleans...
Private m_bUserCancel As Boolean
Private m_bScaleSetOnOpen As Boolean
Private m_bCurrTypeSetOnOpen As Boolean




'=========================================================
'==================  (Private) EVENTS  ===================
'=========================================================
```

```
'==========================================================================
Private Sub UserForm_Initialize()
'[VBA QUIRK:  Errors in Intialize, Activate, and Terminate event handlers _
cannot be handled by procedures which create or destroy instances of the _
class. If any such error-handling IS required, create a separate public _
procedure - e.g., the Initialize procedure, below.]

End Sub
'==========================================================================
Private Sub UserForm_Activate()
'Apparently we need to refresh the RowSource property of combo boxes whenever _
a form is activated...
Dim rngRowSrc As Range
Set rngRowSrc = ThisWorkbook.Names("vallstdoScaleInclVaries").RefersToRange
cmbScale.RowSource = rngRowSrc.Address(External:=True)
End Sub
'==========================================================================
Private Sub UserForm_Terminate()
'[VBA QUIRK:  Errors in Intialize, Activate, and Terminate event handlers _
cannot be handled by procedures which create or destroy instances of the _
class. If any such error-handling IS required, create a separate public _
procedure - e.g., the Initialize procedure, below.]

'Since there's really nothing useful an error-handler can do at this point...
On Error Resume Next

End Sub
'==========================================================================
Private Sub txtDataSrc_Change()
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "txtDataSrc_Change()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

On Error GoTo ErrorHandler

'CODE HERE...
TrapForEntriesInPairedFields _
        TxtBoxCtrlName:=txtDataSrc.name, _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
m_bUserCancel = False
Exit Sub

ErrorHandler:
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'==========================================================================
Private Sub txtFunc_Change()
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "txtFunc_Change()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

On Error GoTo ErrorHandler
```

```
'CODE HERE...
TrapForEntriesInPairedFields _
    TxtBoxCtrlName:=txtFunc.name, _
    RanOkRESULT:=bCalledProcedureRanOk
If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
m_bUserCancel = False
Exit Sub

ErrorHandler:
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'================================================================================
Private Sub txtLdgr_Change()
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "txtLdgr_Change()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

On Error GoTo ErrorHandler

'CODE HERE...
TrapForEntriesInPairedFields _
    TxtBoxCtrlName:=txtLdgr.name, _
    RanOkRESULT:=bCalledProcedureRanOk
If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
m_bUserCancel = False
Exit Sub

ErrorHandler:
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'================================================================================
Private Sub txtLglEnt_Change()
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "txtLglEnt_Change()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
```

```
On Error GoTo ErrorHandler

'CODE HERE...
TrapForEntriesInPairedFields _
    TxtBoxCtrlName:=txtLglEnt.name, _
    RanOkRESULT:=bCalledProcedureRanOk
If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
m_bUserCancel = False
Exit Sub

ErrorHandler:
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'================================================================================
Private Sub txtMgdBus_Change()
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "txtMgdBus_Change()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

On Error GoTo ErrorHandler

'CODE HERE...
TrapForEntriesInPairedFields _
    TxtBoxCtrlName:=txtMgdBus.name, _
    RanOkRESULT:=bCalledProcedureRanOk
If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
m_bUserCancel = False
Exit Sub

ErrorHandler:
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'================================================================================
Private Sub txtMVAcct_Change()
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "txtMVAcct()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
```

```
On Error GoTo ErrorHandler

'CODE HERE...
TrapForEntriesInPairedFields _
    TxtBoxCtrlName:=txtMVAcct.name, _
    RanOkRESULT:=bCalledProcedureRanOk
If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
m_bUserCancel = False
Exit Sub

ErrorHandler:
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'================================================================================
Private Sub txtOpLines_Change()
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "txtOpLines_Change()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

On Error GoTo ErrorHandler

'CODE HERE...
TrapForEntriesInPairedFields _
    TxtBoxCtrlName:=txtOpLines.name, _
    RanOkRESULT:=bCalledProcedureRanOk
If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
m_bUserCancel = False
Exit Sub

ErrorHandler:
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'================================================================================
Private Sub txtOrigGeog_Change()
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "txtOrigGeog_Change()"
Dim bCalledProcedureRanOk As Boolean
```

```
'Operating code declarations...

On Error GoTo ErrorHandler

'CODE HERE...
TrapForEntriesInPairedFields _
    TxtBoxCtrlName:=txtOrigGeog.name, _
    RanOkRESULT:=bCalledProcedureRanOk
If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
m_bUserCancel = False
Exit Sub

ErrorHandler:
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'================================================================================
Private Sub txtPer_Change()
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "txtPer_Change()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

On Error GoTo ErrorHandler

'CODE HERE...
TrapForEntriesInPairedFields _
    TxtBoxCtrlName:=txtPer.name, _
    RanOkRESULT:=bCalledProcedureRanOk
If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
m_bUserCancel = False
Exit Sub

ErrorHandler:
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'================================================================================
Private Sub txtProd_Change()
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "txtProd_Change()"
```

```
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

On Error GoTo ErrorHandler

'CODE HERE...
TrapForEntriesInPairedFields _
    TxtBoxCtrlName:=txtProd.name, _
    RanOkRESULT:=bCalledProcedureRanOk
If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
m_bUserCancel = False
Exit Sub

ErrorHandler:
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'============================================================================
Private Sub txtScenario_Change()
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "txtScenario_Change()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

On Error GoTo ErrorHandler

'CODE HERE...
TrapForEntriesInPairedFields _
    TxtBoxCtrlName:=txtScenario.name, _
    RanOkRESULT:=bCalledProcedureRanOk
If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
m_bUserCancel = False
Exit Sub

ErrorHandler:
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'============================================================================
Private Sub txtYr_Change()
'Error-handling declarations...
```

```
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "txtYr_Change()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

On Error GoTo ErrorHandler

'CODE HERE...
TrapForEntriesInPairedFields _
    TxtBoxCtrlName:=txtYr.name, _
    RanOkRESULT:=bCalledProcedureRanOk
If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
m_bUserCancel = False
Exit Sub

ErrorHandler:
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'================================================================================
Private Sub btnOK_Click()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "btnOK_Click()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

On Error GoTo ErrorHandler

If g_bStepThruMode Then Stop
g_cFormsHandler.SetFileDimenDataFmFormMaestro bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
m_bUserCancel = False
Exit Sub

ErrorHandler:
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'================================================================================
Private Sub btnCancel_Click()
```

```
    'NOTE 1:  MSDN has some notes on creating and setting the following _
private boolean.  I still have no idea about how to work with it...
'm_bUserCancel = True
    'NOTE 2:  We can't use our centralized error-trapping with a call to _
g_cFormsHandler.UserWantsToCxlCloseForm m_sFORM_NAME.  That procedure may _
wind up Unload-ing this form.  If so, there would be no object to hold our _
typical bCalledProcedureRanOK boolean.  As such, that value would flip to _
its default of FALSE.  Remaining code in this btnCancel_Click() WOULD _
continue to run.  However, with the boolean flipping to FALSE we would _
always generate an error.  Therefore, as noted in the _
g_cFormsHandler.UserWantsToCxlCloseForm m_sFORM_NAME sub, we effectively _
treat THAT sub as our entry point...
g_cFormsHandler.UserWantsToCxlCloseForm m_sFORM_NAME
End Sub
'================================================================================
Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
'......................
'MSDN on...
'Cancel:  "Setting this argument to any value other than 0 [i.e., other than _
FALSE, i.e., "Setting this argument to TRUE"...] stops the QueryClose event in _
all loaded user forms and prevents the UserForm and application from closing."
'CloseMode:  "A value or constant indicating the cause of the QueryClose event"
'......................

'NOTE:  vbFormControlMenu ==> _
    The user has chosen the Close command from the Control menu on the UserForm.
'NOTE:  False === 0

'Route any X-close button calls through the btnCancel_Click procedure...
If CloseMode = vbFormControlMenu Then
    g_cMsgBoxHandler.SetPlsUseOKCxlBtnsMsg
    g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbExclamation
    Cancel = True 'i.e., -1
End If
End Sub



'============================================================
'==================  (Private) PROCEDURES  ==================
'============================================================
'================================================================================
'================================================================================
Private Sub TagAllTxtBoxAndRefEditControls(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "TagAllTxtBoxAndRefEditControls()"
Const bSUB_IS_ENTRY_PT As Boolean = False
'Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Me.txtDataSrc.Tag = g_sDIMEN_DATA_SRC
Me.txtFunc.Tag = g_sDIMEN_FUNC
Me.txtLdgr.Tag = g_sDIMEN_LDGR
Me.txtLglEnt.Tag = g_sDIMEN_LGL_ENT
Me.txtMgdBus.Tag = g_sDIMEN_MGD_BUS
Me.txtMVAcct.Tag = g_sDIMEN_MV_ACCT
Me.txtOpLines.Tag = g_sDIMEN_OPER_LNS
Me.txtOrigGeog.Tag = g_sDIMEN_ORIG_GEOG
Me.txtPer.Tag = g_sDIMEN_PER
Me.txtProd.Tag = g_sDIMEN_PROD
Me.txtScenario.Tag = g_sDIMEN_SCENAR
Me.txtYr.Tag = g_sDIMEN_YR

Me.refDataSrc.Tag = g_sDIMEN_DATA_SRC
Me.refFunc.Tag = g_sDIMEN_FUNC
Me.refLedger.Tag = g_sDIMEN_LDGR
Me.refLglEnt.Tag = g_sDIMEN_LGL_ENT
```

```
Me.refMngdBus.Tag = g_sDIMEN_MGD_BUS
Me.refMVAcct.Tag = g_sDIMEN_MV_ACCT
Me.refOpLines.Tag = g_sDIMEN_OPER_LNS
Me.refOrigGeog.Tag = g_sDIMEN_ORIG_GEOG
Me.refPer.Tag = g_sDIMEN_PER
Me.refProd.Tag = g_sDIMEN_PROD
Me.refScenario.Tag = g_sDIMEN_SCENAR
Me.refYr.Tag = g_sDIMEN_YR

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
            ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'================================================================================
Private Sub PopulateRefEditTextBoxDictionary(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "PopulateRefEditTextBoxDictionary()"
Const bSUB_IS_ENTRY_PT As Boolean = False
'Operating code declarations...
Dim sTxtBoxTag As String
Dim ctrlTxtBox As MSForms.TextBox
Dim ctrlRefEdit As MSForms.control

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
For Each ctrlTxtBox In m_collTextBoxesByCtrlNm
    'If g_bStepThruMode Then Stop
    sTxtBoxTag = ctrlTxtBox.Tag
    'Now pair with matching RefEdit control...
    For Each ctrlRefEdit In m_collRefEditsByCtrlNm
        If StrComp(ctrlRefEdit.Tag, sTxtBoxTag) = 0 Then
            'If g_bStepThruMode Then Stop
            m_dictRefEditByTxtBox.Add _
                keyCtrlName:=ctrlTxtBox.name, _
                valCtrlName:=ctrlRefEdit.name
        End If
    Next ctrlRefEdit
Next ctrlTxtBox

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
            ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
```

```
            IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'============================================================================
Private Sub TrapForEntriesInPairedFields(ByRef TxtBoxCtrlName As String, _
    ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "TrapForEntriesInPairedFields()"
Const bSUB_IS_ENTRY_PT As Boolean = False
'Operating code declarations...
Dim sRefEditCtrlNm As String

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
sRefEditCtrlNm = m_dictRefEditByTxtBox.Item(TxtBoxCtrlName)

If StrComp(Trim(Me.Controls(TxtBoxCtrlName).value), vbNullString) <> 0 Then
    With Me.Controls(sRefEditCtrlNm)
        .value = vbNullString
        .BackColor = g_lFIELD_COLOR_DISABLED
        .enabled = False
    End With
Else '...text box control is now empty
    With Me.Controls(sRefEditCtrlNm)
        .enabled = True
        .BackColor = g_lFIELD_COLOR_ENABLED
    End With
End If

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'============================================================================
Private Sub PopulateFormControlCollections(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "PopulateFormControlCollections()"
Const bSUB_IS_ENTRY_PT As Boolean = False
'''Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim objFormCtrl As Object
Dim sTypNm As String

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
```

```
frmWbkMandDimens - 12

RanOkRESULT = True

'CODE HERE...
For Each objFormCtrl In Me.Controls
    sTypNm = Trim(TypeName(objFormCtrl))
    If StrComp(sTypNm, "TextBox") = 0 Then
        m_collTextBoxesByCtrlNm.Add objFormCtrl
        m_collTxtBoxByDimenNm.Add objFormCtrl
    End If
    If StrComp(sTypNm, "RefEdit") = 0 Then
        m_collRefEditsByCtrlNm.Add objFormCtrl
        m_collRefEditByDimNm.Add objFormCtrl
    End If
Next objFormCtrl

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub




'******************************************************************************
'******************************************************************************
'*****************************  PUBLIC MEMBERS  *******************************
'******************************************************************************
'******************************************************************************

'============================================================
'======================  PROPERTIES  ========================
'============================================================
'==============================================================================
'Property UserCancel (read-only)...
Property Get UserCancel() As Boolean
    UserCancel = m_bUserCancel
End Property
'Property TextBoxesByCtrlNm (read-only)...
Property Get TextBoxesByCtrlNm() As ByNmFormControls
    Set TextBoxesByCtrlNm = m_collTextBoxesByCtrlNm
End Property
'Property RefEditsByCtrlNm (read-only)...
Property Get RefEditsByCtrlNm() As ByNmFormControls
    Set RefEditsByCtrlNm = m_collRefEditsByCtrlNm
End Property
'Property TextBoxesByDimenNm (read-only)...
Property Get TextBoxesByDimenNm() As ByDimenNmFrmCtrls
    Set TextBoxesByDimenNm = m_collTxtBoxByDimenNm
End Property
'Property RefEditsByDimenNm (read-only)...
Property Get RefEditsByDimenNm() As ByDimenNmFrmCtrls
    Set RefEditsByDimenNm = m_collRefEditByDimNm
End Property
```

```
'Property DimensPopulatedOnOpen (read-only)...
Property Get DimensPopulatedOnOpen() As Dimensions
    Set DimensPopulatedOnOpen = m_collDimensPopulatedOnOpen
End Property
'Property ScaleSetOnOpen...
Property Get ScaleSetOnOpen() As Boolean
    ScaleSetOnOpen = m_bScaleSetOnOpen
End Property
Property Let ScaleSetOnOpen(value As Boolean)
    m_bScaleSetOnOpen = value
End Property
'Property CurrTypeSetOnOpen...
Property Get CurrTypeSetOnOpen() As Boolean
    CurrTypeSetOnOpen = m_bCurrTypeSetOnOpen
End Property
Property Let CurrTypeSetOnOpen(value As Boolean)
    m_bCurrTypeSetOnOpen = value
End Property


'============================================================
'=================  (Public) PROCEDURES  ==================
'============================================================
'============================================================================
Public Sub Initialize( _
        ByVal DictCompareMeth As CompareMethod, _
        ByRef RanOkRESULT As Boolean)
'[NOTE:  Use this procedure whenever there is start-up code which might _
possibly fail.  Put such code here, which is a procedure which must be _
called expicitly, because, as noted above, errors which might occur in _
the UserForm_Initialize() method CANNOT be trapped by a calling procedure.]
'Error-handling declarations...
Const sSOURCE As String = "Initialize()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim ctrlForm As MSForms.control

On Error GoTo ErrorHandler
'Assume success until an error is encountered...
RanOkRESULT = True

Set m_dictRefEditByTxtBox = New CtrlNmByCtrlNm
Set m_collTextBoxesByCtrlNm = New ByNmFormControls
Set m_collRefEditsByCtrlNm = New ByNmFormControls
Set m_collTxtBoxByDimenNm = New ByDimenNmFrmCtrls
Set m_collRefEditByDimNm = New ByDimenNmFrmCtrls
Set m_collDimensPopulatedOnOpen = New Dimensions

TagAllTxtBoxAndRefEditControls RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

'If g_bStepThruMode Then Stop
PopulateFormControlCollections RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

PopulateRefEditTextBoxDictionary RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
Exit Sub

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
```

```
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'============================================================================
'============================================================================
'============================================================================
'============================================================================
'============================================================================
'============================================================================
```

frmWshsToImport - 1

```
'.....................................................................................
'.......        Developer:              William H. White (consultant)
'.......        With:                   TEKsystems Inc.
'.......                                www.teksystems.com
'.......        For:                    AIG
'.......                                Financial Information Systems
'.......                                1 NY Plaza, 15th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         9/10/2013
'.......        Proj finished:          11/21/2013
'.......        Proj revised:           1/15/2014
'.....................................................................................
'.....................................................................................

'.....................................................................................
'.........................  Form-level COMMENTS  .........................
'Commentary...
'.....................................................................................
'.....................................................................................



'*************************************************************************************
'*************************************************************************************
'*************************  FORM-LEVEL DECLARATIONS  *************************
'*************************************************************************************
'*************************************************************************************
Option Explicit
Option Compare Binary '...use cAsE senSITive string comparisons




'*************************************************************************************
'*************************************************************************************
'*******************************  PRIVATE MEMBERS  *******************************
'*************************************************************************************
'*************************************************************************************

'**************************************
'FORM-WIDE CONSTANTS
'**************************************
'Strings...
Private Const m_sFORM_NAME As String = "frmWshsToImport"


'**************************************
'MODULE/Form-WIDE VARIABLES
'**************************************
'Booleans...
Private m_bUserCancel As Boolean




'============================================================
'==================  (Private) EVENTS  ====================
'============================================================
'=============================================================================
Private Sub UserForm_Initialize()
'[VBA QUIRK:  Errors in Intialize, Activate, and Terminate event handlers _
cannot be handled by procedures which create or destroy instances of the _
class. If any such error-handling IS required, create a separate public _
procedure - e.g., the Initialize procedure, below.]

End Sub
'=============================================================================
```

```
Private Sub UserForm_Terminate()
'[VBA QUIRK:  Errors in Intialize, Activate, and Terminate event handlers _
cannot be handled by procedures which create or destroy instances of the _
class. If any such error-handling IS required, create a separate public _
procedure - e.g., the Initialize procedure, below.]

'Since there's really nothing useful an error-handler can do at this point...
On Error Resume Next

End Sub
'================================================================================
Private Sub btnOK_Click()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "btnOK_Click()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

On Error GoTo ErrorHandler

g_cFormsHandler.UserOKClosesWshsToImportForm RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
m_bUserCancel = False
Exit Sub

ErrorHandler:
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'================================================================================
Private Sub btnCancel_Click()
    'NOTE 1:  MSDN has some notes on creating and setting the following _
private boolean.  I still have no idea about how to work with it...
'm_bUserCancel = True
    'NOTE 2:  We can't use our centralized error-trapping with a call to _
g_cFormsHandler.UserWantsToCxlCloseForm m_sFORM_NAME.  That procedure may _
wind up Unload-ing this form.  If so, there would be no object to hold our _
typical bCalledProcedureRanOK boolean.  As such, that value would flip to _
its default of FALSE.  Remaining code in this btnCancel_Click() WOULD _
continue to run.  However, with the boolean flipping to FALSE we would _
always generate an error.  Therefore, as noted in the _
g_cFormsHandler.UserWantsToCxlCloseForm m_sFORM_NAME sub, we effectively _
treat THAT sub as our entry point...
g_cFormsHandler.UserWantsToCxlCloseForm m_sFORM_NAME
End Sub
'================================================================================
Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
'......................
'MSDN on...
'Cancel:  "Setting this argument to any value other than 0 [i.e., other than _
FALSE, i.e., "Setting this argument to TRUE"...] stops the QueryClose event in _
all loaded user forms and prevents the UserForm and application from closing."
'CloseMode:  "A value or constant indicating the cause of the QueryClose event"
'......................

'NOTE:  vbFormControlMenu ==> _
```

```
frmWshsToImport - 3

    The user has chosen the Close command from the Control menu on the UserForm.
'NOTE:  False === 0

'Route any X-close button calls through the btnCancel_Click procedure...
If CloseMode = vbFormControlMenu Then
    g_cMsgBoxHandler.SetPlsUseOKCxlBtnsMsg
    g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbExclamation
    Cancel = True 'i.e., -1
End If
End Sub



'*****************************************************************************
'*****************************************************************************
'*****************************  PUBLIC MEMBERS  ******************************
'*****************************************************************************
'*****************************************************************************

'============================================================
'======================  PROPERTIES  ========================
'============================================================
'============================================================================
'Property UserCancel (read-only)...
Property Get UserCancel() As Boolean
    UserCancel = m_bUserCancel
End Property

'============================================================================
'============================================================================
'============================================================================
'============================================================================
'============================================================================
'============================================================================
```

```
'.......        Developer:              William H. White (consultant)
'.......        With:                   TEKsystems Inc.
'.......                                www.teksystems.com
'.......        For:                    AIG
'.......                                Financial Information Systems
'.......                                1 NY Plaza, 15th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         9/13/2013
'.......        Proj finished:          11/21/2013
'.......        Proj revised:           1/15/2014
'.....................................................................................
'.....................................................................................

'.....................................................................................
'.........................  Form-level COMMENTS  .....................................
'Commentary...
'.....................................................................................
'.....................................................................................



'*************************************************************************************
'*************************************************************************************
'*************************  FORM-LEVEL DECLARATIONS  *************************************
'*************************************************************************************
'*************************************************************************************
Option Explicit
Option Compare Binary '...use cAsE senSITive string comparisons




'*************************************************************************************
'*************************************************************************************
'*******************************  PRIVATE MEMBERS  ***********************************
'*************************************************************************************
'*************************************************************************************

'**************************************
'FORM-WIDE CONSTANTS
'**************************************
'Strings...
Private Const m_sFORM_NAME As String = "frmWshTypeDimens"

'Bytes...
Private Const m_yCMB_BOX_TTL_NO_COLS = 2
Private Const m_yCMB_BOX_WSH_TYPE_NM_COL = 0
Private Const m_yCMB_BOX_WSH_DEFINED_COL = 1


'**************************************
'MODULE/Form-WIDE VARIABLES
'**************************************
'Collections...
Private m_collDimenLabelsByCtrlNm As ByNmFormControls
Private m_collFileDimenLblsByCtrlNm As ByNmFormControls
Private m_collAllRefEdsByCtrlNm As ByNmFormControls
Private m_collWshDimenRefEdsByCtrlNm As ByNmFormControls
Private m_collRowDimenRefEdsByCtrlNm As ByNmFormControls
Private m_collColDimenRefEdsByCtrlNm As ByNmFormControls

Private m_collDimenLblsByDimenNm As ByDimenNmFrmCtrls
Private m_collFileDimenLblsByDimenNm As ByDimenNmFrmCtrls
Private m_collWshDimenRefEdsByDimenNm As ByDimenNmFrmCtrls
Private m_collColDimenRefEdsByDimenNm As ByDimenNmFrmCtrls
Private m_collRowDimenRefEdsByDimenNm As ByDimenNmFrmCtrls
```

```
frmWshTypeDimens - 2

'Strings...
Private m_sWshTypeNm As String
Private m_sWshTypeNmColHdg  As String

'Booleans...
Private m_bUserCancel As Boolean
Private m_bIgnoreWshTypeCmbBoxChng As Boolean




'============================================================
'==================  (Private) EVENTS  ====================
'============================================================
'================================================================================
Private Sub UserForm_Initialize()
'[VBA QUIRK:  Errors in Intialize, Activate, and Terminate event handlers _
cannot be handled by procedures which create or destroy instances of the _
class. If any such error-handling IS required, create a separate public _
procedure - e.g., the Initialize procedure, below.]

End Sub
'================================================================================
Private Sub UserForm_Activate()
'Apparently we need to refresh the RowSource property of combo boxes whenever _
a form is activated...
Dim rngRowSrc As Range
Set rngRowSrc = ThisWorkbook.Names("vallstdoScale").RefersToRange
cmbScale.RowSource = rngRowSrc.Address(External:=True)
End Sub
'================================================================================
Private Sub UserForm_Terminate()
'[VBA QUIRK:  Errors in Intialize, Activate, and Terminate event handlers _
cannot be handled by procedures which create or destroy instances of the _
class. If any such error-handling IS required, create a separate public _
procedure - e.g., the Initialize procedure, below.]

'Since there's really nothing useful an error-handler can do at this point...
On Error Resume Next

End Sub
'================================================================================
Private Sub cmbWshType_Change()
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "cmbWshType_Change()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim bLeaveFormOpen As Boolean
Dim sSelectedWshType As String

On Error GoTo ErrorHandler
If g_bStepThruMode Then Stop

If m_bIgnoreWshTypeCmbBoxChng Then GoTo ExitPoint
If Me.cmbWshType.ListIndex = -1 Then GoTo ExitPoint '...nothing selected

sSelectedWshType = _
        Trim(Me.cmbWshType.List(Me.cmbWshType.ListIndex, _
        m_yCMB_BOX_WSH_TYPE_NM_COL))

'Trap for user picking the header row...
If StrComp(sSelectedWshType, m_sWshTypeNmColHdg) = 0 Then
    g_cMsgBoxHandler.SetPickValidWshTypeMsg
    g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbExclamation
    GoTo ExitPoint
End If

'Trap for user picking the WshType he is already on...
If StrComp(sSelectedWshType, Me.WshTypeNm, vbTextCompare) = 0 Then _
        GoTo ExitPoint
'Me.cmbWshType.List
```

```
frmWshTypeDimens - 3

If g_bStepThruMode Then Stop
Me.Hide
g_cFormsHandler.OpenWshTypeDimenIDForm _
        OpenedFmAnotherWTDForm:=True, _
        RanOkRESULT:=bCalledProcedureRanOk, _
        WshTypeNm:=sSelectedWshType
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
m_bUserCancel = False
Exit Sub

ErrorHandler:
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'============================================================================
Private Sub btnOK_Click()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "btnTxtOK_Click()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim sWshType As String

On Error GoTo ErrorHandler

If g_bStepThruMode Then Stop
sWshType = _
    Me.cmbWshType.List(Me.cmbWshType.ListIndex, m_yCMB_BOX_WSH_TYPE_NM_COL)
sWshType = Trim(sWshType)
g_cFormsHandler.SetWshTypeDimenDataFmFormMaestro _
        RanOkRESULT:=bCalledProcedureRanOk, _
        WshTypeNm:=sWshType
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
m_bUserCancel = False
Exit Sub

ErrorHandler:
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'============================================================================
```

```
Private Sub btnCancel_Click()
    'NOTE 1:  MSDN has some notes on creating and setting the following _
private boolean.  I still have no idea about how to work with it...
'm_bUserCancel = True
    'NOTE 2:  We can't use our centralized error-trapping with a call to _
g_cFormsHandler.UserWantsToCxlCloseForm m_sFORM_NAME.  That procedure may _
wind up Unload-ing this form.  If so, there would be no object to hold our _
typical bCalledProcedureRanOK boolean.  As such, that value would flip to _
its default of FALSE.  Remaining code in this btnCancel_Click() WOULD _
continue to run.  However, with the boolean flipping to FALSE we would _
always generate an error.  Therefore, as noted in the _
g_cFormsHandler.UserWantsToCxlCloseForm m_sFORM_NAME sub, we effectively _
treat THAT sub as our entry point...
g_cFormsHandler.UserWantsToCxlCloseForm m_sFORM_NAME, Me.WshTypeNm
End Sub
'================================================================================
Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
'......................
'MSDN on...
'Cancel:  "Setting this argument to any value other than 0 [i.e., other than _
FALSE, i.e., "Setting this argument to TRUE"...] stops the QueryClose event in _
all loaded user forms and prevents the UserForm and application from closing."
'CloseMode:  "A value or constant indicating the cause of the QueryClose event"
'......................

'NOTE:  vbFormControlMenu ==> _
    The user has chosen the Close command from the Control menu on the UserForm.
'NOTE:  False === 0

'Route any X-close button calls through the btnCancel_Click procedure...
If CloseMode = vbFormControlMenu Then
    g_cMsgBoxHandler.SetPlsUseOKCxlBtnsMsg
    g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbExclamation
    Cancel = True 'i.e., -1
End If
End Sub



'============================================================
'==================  (Private) PROCEDURES  ==================
'============================================================
'================================================================================
'================================================================================
Private Sub TagAllFormControls(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "TagAllFormControls()"
Const bSUB_IS_ENTRY_PT As Boolean = False
'Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Me.lblDataSrc.Tag = g_sDIMEN_DATA_SRC
Me.lblFileDimDataSrc.Tag = g_sDIMEN_DATA_SRC
Me.refWshDimDataSrc.Tag = g_sDIMEN_DATA_SRC
Me.refRowDimDataSrc.Tag = g_sDIMEN_DATA_SRC
Me.refColDimDataSrc.Tag = g_sDIMEN_DATA_SRC

Me.lblFunc.Tag = g_sDIMEN_FUNC
Me.lblFileDimFunc.Tag = g_sDIMEN_FUNC
Me.refWshDimFunc.Tag = g_sDIMEN_FUNC
Me.refRowDimFunc.Tag = g_sDIMEN_FUNC
Me.refColDimFunc.Tag = g_sDIMEN_FUNC

Me.lblLdgr.Tag = g_sDIMEN_LDGR
Me.lblFileDimLdgr.Tag = g_sDIMEN_LDGR
Me.refWshDimLdgr.Tag = g_sDIMEN_LDGR
Me.refRowDimLdgr.Tag = g_sDIMEN_LDGR
```

```
Me.refColDimLdgr.Tag = g_sDIMEN_LDGR

Me.lblLglEnt.Tag = g_sDIMEN_LGL_ENT
Me.lblFileDimLglEnt.Tag = g_sDIMEN_LGL_ENT
Me.refWshDimLglEnt.Tag = g_sDIMEN_LGL_ENT
Me.refRowDimLglEnt.Tag = g_sDIMEN_LGL_ENT
Me.refColDimLglEnt.Tag = g_sDIMEN_LGL_ENT

Me.lblMgdBus.Tag = g_sDIMEN_MGD_BUS
Me.lblFileDimMgdBus.Tag = g_sDIMEN_MGD_BUS
Me.refWshDimMgdBus.Tag = g_sDIMEN_MGD_BUS
Me.refRowDimMgdBus.Tag = g_sDIMEN_MGD_BUS
Me.refColDimMgdBus.Tag = g_sDIMEN_MGD_BUS

Me.lblMVAcct.Tag = g_sDIMEN_MV_ACCT
Me.lblFileDimMVAcct.Tag = g_sDIMEN_MV_ACCT
Me.refWshDimMVAcct.Tag = g_sDIMEN_MV_ACCT
Me.refRowDimMVAcct.Tag = g_sDIMEN_MV_ACCT
Me.refColDimMVAcct.Tag = g_sDIMEN_MV_ACCT

Me.lblOpLines.Tag = g_sDIMEN_OPER_LNS
Me.lblFileDimOpLines.Tag = g_sDIMEN_OPER_LNS
Me.refWshDimOpLines.Tag = g_sDIMEN_OPER_LNS
Me.refRowDimOpLines.Tag = g_sDIMEN_OPER_LNS
Me.refColDimOpLines.Tag = g_sDIMEN_OPER_LNS

Me.lblOrigGeog.Tag = g_sDIMEN_ORIG_GEOG
Me.lblFileDimOrigGeog.Tag = g_sDIMEN_ORIG_GEOG
Me.refWshDimOrigGeog.Tag = g_sDIMEN_ORIG_GEOG
Me.refRowDimOrigGeog.Tag = g_sDIMEN_ORIG_GEOG
Me.refColDimOrigGeog.Tag = g_sDIMEN_ORIG_GEOG

Me.lblPeriod.Tag = g_sDIMEN_PER
Me.lblFileDimPer.Tag = g_sDIMEN_PER
Me.refWshDimPer.Tag = g_sDIMEN_PER
Me.refRowDimPer.Tag = g_sDIMEN_PER
Me.refColDimPer.Tag = g_sDIMEN_PER

Me.lblProd.Tag = g_sDIMEN_PROD
Me.lblFileDimProd.Tag = g_sDIMEN_PROD
Me.refWshDimProd.Tag = g_sDIMEN_PROD
Me.refRowDimProd.Tag = g_sDIMEN_PROD
Me.refColDimProd.Tag = g_sDIMEN_PROD

Me.lblScen.Tag = g_sDIMEN_SCENAR
Me.lblFileDimScen.Tag = g_sDIMEN_SCENAR
Me.refWshDimScen.Tag = g_sDIMEN_SCENAR
Me.refRowDimScen.Tag = g_sDIMEN_SCENAR
Me.refColDimScen.Tag = g_sDIMEN_SCENAR

Me.lblYear.Tag = g_sDIMEN_YR
Me.lblFileDimYr.Tag = g_sDIMEN_YR
Me.refWshDimYr.Tag = g_sDIMEN_YR
Me.refRowDimYr.Tag = g_sDIMEN_YR
Me.refColDimYr.Tag = g_sDIMEN_YR

Me.lblScale.Tag = g_sDIMEN_SCALE
Me.lblFileDimScale.Tag = g_sDIMEN_SCALE
Me.cmbScale.Tag = g_sDIMEN_SCALE

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
            ProcedureName:=sSOURCE, _
```

```
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'================================================================================
Private Sub PopulateFormControlCollections(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "PopulateFormControlCollections()"
Const bSUB_IS_ENTRY_PT As Boolean = False
'Operating code declarations...
Dim objFormCtrl As Object

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Set m_collDimenLabelsByCtrlNm = New ByNmFormControls
m_collDimenLabelsByCtrlNm.Add Me.lblAmtType
m_collDimenLabelsByCtrlNm.Add Me.lblDataSrc
m_collDimenLabelsByCtrlNm.Add Me.lblFunc
m_collDimenLabelsByCtrlNm.Add Me.lblLdgr
m_collDimenLabelsByCtrlNm.Add Me.lblLglEnt
m_collDimenLabelsByCtrlNm.Add Me.lblMgdBus
m_collDimenLabelsByCtrlNm.Add Me.lblMVAcct
m_collDimenLabelsByCtrlNm.Add Me.lblOpLines
m_collDimenLabelsByCtrlNm.Add Me.lblOrigGeog
m_collDimenLabelsByCtrlNm.Add Me.lblPeriod
m_collDimenLabelsByCtrlNm.Add Me.lblProd
m_collDimenLabelsByCtrlNm.Add Me.lblScen
m_collDimenLabelsByCtrlNm.Add Me.lblYear

For Each objFormCtrl In Me.DimensionLabelsByCtrlNm
    m_collDimenLblsByDimenNm.Add objFormCtrl
Next objFormCtrl

For Each objFormCtrl In Me.framFileDimens.Controls
    m_collFileDimenLblsByCtrlNm.Add objFormCtrl
    m_collFileDimenLblsByDimenNm.Add objFormCtrl
Next objFormCtrl

For Each objFormCtrl In Me.framWshDimens.Controls
    m_collAllRefEdsByCtrlNm.Add objFormCtrl
    m_collWshDimenRefEdsByCtrlNm.Add objFormCtrl
    m_collWshDimenRefEdsByDimenNm.Add objFormCtrl
Next objFormCtrl

For Each objFormCtrl In Me.framRowDimens.Controls
    m_collAllRefEdsByCtrlNm.Add objFormCtrl
    m_collRowDimenRefEdsByCtrlNm.Add objFormCtrl
    m_collRowDimenRefEdsByDimenNm.Add objFormCtrl
Next objFormCtrl

For Each objFormCtrl In Me.framColDimens.Controls
    m_collAllRefEdsByCtrlNm.Add objFormCtrl
    m_collColDimenRefEdsByCtrlNm.Add objFormCtrl
    m_collColDimenRefEdsByDimenNm.Add objFormCtrl
Next objFormCtrl

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
```

```
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub




'*******************************************************************************
'*******************************************************************************
'*****************************  PUBLIC MEMBERS  ********************************
'*******************************************************************************
'*******************************************************************************

'=============================================================
'======================  PROPERTIES  =========================
'=============================================================
'==============================================================================
'Property UserCancel (read-only)...
Property Get UserCancel() As Boolean
    UserCancel = m_bUserCancel
End Property
'Property WshTypeNm (read-only)...
Property Get WshTypeNm() As String
    WshTypeNm = m_sWshTypeNm
End Property
'Property DimensionLabelsByCtrlNm (read-only)...
Property Get DimensionLabelsByCtrlNm() As ByNmFormControls
Set DimensionLabelsByCtrlNm = m_collDimenLabelsByCtrlNm
End Property
'Property FileDimenLabelsByCtrlNm (read-only)...
Property Get FileDimenLabelsByCtrlNm() As ByNmFormControls
    Set FileDimenLabelsByCtrlNm = m_collFileDimenLblsByCtrlNm
End Property
'Property AllRefEditsByCtrlNm (read-only)...
Property Get AllRefEditsByCtrlNm() As ByNmFormControls
    Set AllRefEditsByCtrlNm = m_collAllRefEdsByCtrlNm
End Property
'Property WshDimenRefEditsByCtrlNm (read-only)...
Property Get WshDimenRefEditsByCtrlNm() As ByNmFormControls
    Set WshDimenRefEditsByCtrlNm = m_collWshDimenRefEdsByCtrlNm
End Property
'Property RowDimenRefEditsByCtrlNm (read-only)...
Property Get RowDimenRefEditsByCtrlNm() As ByNmFormControls
    Set RowDimenRefEditsByCtrlNm = m_collRowDimenRefEdsByCtrlNm
End Property
'Property ColDimenRefEditsByCtrlNm (read-only)...
Property Get ColDimenRefEditsByCtrlNm() As ByNmFormControls
    Set ColDimenRefEditsByCtrlNm = m_collColDimenRefEdsByCtrlNm
End Property
'Property DimenLblsByDimenNm (read-only)...
Property Get DimenLblsByDimenNm() As ByDimenNmFrmCtrls
    Set DimenLblsByDimenNm = m_collDimenLblsByDimenNm
End Property
'Property FileDimenLblsByDimenNm (read-only)...
Property Get FileDimenLblsByDimenNm() As ByDimenNmFrmCtrls
    Set FileDimenLblsByDimenNm = m_collFileDimenLblsByDimenNm
End Property
'Property WshDimenRefEditsByDimenNm (read-only)...
Property Get WshDimenRefEditsByDimenNm() As ByDimenNmFrmCtrls
```

```
    Set WshDimenRefEditsByDimenNm = m_collWshDimenRefEdsByDimenNm
End Property
'Property ColDimenRefEditsByDimenNm (read-only)...
Property Get ColDimenRefEditsByDimenNm() As ByDimenNmFrmCtrls
    Set ColDimenRefEditsByDimenNm = m_collColDimenRefEdsByDimenNm
End Property
'Property RowDimenRefEditsByDimenNm (read-only)...
Property Get RowDimenRefEditsByDimenNm() As ByDimenNmFrmCtrls
    Set RowDimenRefEditsByDimenNm = m_collRowDimenRefEdsByDimenNm
End Property
'Property IgnoreChngInWshTypeCmbBox...
Property Get IgnoreChngInWshTypeCmbBox() As Boolean
    IgnoreChngInWshTypeCmbBox = m_bIgnoreWshTypeCmbBoxChng
End Property
Property Let IgnoreChngInWshTypeCmbBox(value As Boolean)
    m_bIgnoreWshTypeCmbBoxChng = value
End Property

'============================================================
'=================  (Public) PROCEDURES  ==================
'============================================================
'================================================================================
Public Sub Initialize( _
        ByVal WshTypeNm As String, _
        ByVal FormCaption As String, _
        ByRef RanOkRESULT As Boolean)
'[NOTE:  Use this procedure whenever there is start-up code which might _
possibly fail.  Put such code here, which is a procedure which must be _
called expicitly, because, as noted above, errors which might occur in _
the UserForm_Initialize() method CANNOT be trapped by a calling procedure.]
'Error-handling declarations...
Const sSOURCE As String = "Initialize()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until an error is encountered...
RanOkRESULT = True

m_sWshTypeNm = Trim(WshTypeNm)

Set m_collDimenLabelsByCtrlNm = New ByNmFormControls
Set m_collFileDimenLblsByCtrlNm = New ByNmFormControls
Set m_collAllRefEdsByCtrlNm = New ByNmFormControls
Set m_collWshDimenRefEdsByCtrlNm = New ByNmFormControls
Set m_collRowDimenRefEdsByCtrlNm = New ByNmFormControls
Set m_collColDimenRefEdsByCtrlNm = New ByNmFormControls

Set m_collDimenLblsByDimenNm = New ByDimenNmFrmCtrls
Set m_collFileDimenLblsByDimenNm = New ByDimenNmFrmCtrls
Set m_collWshDimenRefEdsByDimenNm = New ByDimenNmFrmCtrls
Set m_collRowDimenRefEdsByDimenNm = New ByDimenNmFrmCtrls
Set m_collColDimenRefEdsByDimenNm = New ByDimenNmFrmCtrls

TagAllFormControls RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

'If g_bStepThruMode Then Stop
PopulateFormControlCollections RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

PopulateWshTypeComboBox RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Me.Caption = FormCaption

ExitPoint:
Exit Sub

ErrorHandler:
```

```
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'================================================================================
Public Sub PopulateWshTypeComboBox(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "PopulateWshTypeComboBox()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wsSrcWbkValLists As Worksheet
Dim rngWshTypesTbl As Range
Dim sWshTypeNmFmTbl As String
Dim sWshTypeDefinedFmTbl As String
Dim yWshTypeNmIndex As Byte
Dim i As Byte

On Error GoTo ErrorHandler
'Assume success until an error is encountered...
RanOkRESULT = True

If g_bStepThruMode Then Stop
'Stop
'Without this little flag we go into an infinite loop of combobox _
change events when initializing the form...
m_bIgnoreWshTypeCmbBoxChng = True

Set wsSrcWbkValLists = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
Set rngWshTypesTbl = wsSrcWbkValLists.Range("valtbldodynWshTypeInfo")
'm_sWshTypeNmColHdg = Trim(rngWshTypesTbl.Cells(1, g_yWSH_TYPE_TBL_NM_COL))

Me.cmbWshType.ColumnCount = m_yCMB_BOX_TTL_NO_COLS
Me.cmbWshType.BoundColumn = m_yCMB_BOX_WSH_TYPE_NM_COL
Me.cmbWshType.Locked = False
Me.cmbWshType.MatchRequired = True

For i = 1 To rngWshTypesTbl.Rows.Count
    sWshTypeNmFmTbl = Trim(rngWshTypesTbl.Cells(i, g_yWSH_TYPE_TBL_NM_COL))
    sWshTypeDefinedFmTbl = rngWshTypesTbl.Cells(i, g_yWSH_TYPE_TBL_DEFINED_COL)
    'NOTE:  rows & columns in combo box are zero-based...
    Me.cmbWshType.AddItem sWshTypeNmFmTbl
    Me.cmbWshType.List(i - 1, g_yWSH_TYPE_TBL_DEFINED_COL - 1) = _
            sWshTypeDefinedFmTbl
    If StrComp(Trim(WshTypeNm), sWshTypeNmFmTbl) = 0 Then _
            yWshTypeNmIndex = i - 1
Next i

'Highlight the current WshType we are working on...
Me.cmbWshType.ListIndex = yWshTypeNmIndex

m_bIgnoreWshTypeCmbBoxChng = False '...reset

ExitPoint:
Exit Sub

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
```

```
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub


'================================================================================
'================================================================================
'================================================================================
'================================================================================
'================================================================================
'================================================================================
```

```
frmWshTypePicker - 1

'.......        Developer:              William H. White (consultant)
'.......        With:                   TEKsystems Inc.
'.......                                www.teksystems.com
'.......        For:                    AIG
'.......                                Financial Information Systems
'.......                                1 NY Plaza, 15th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         9/21/2013
'.......        Proj finished:          11/21/2013
'.......        Proj revised:           1/15/2014
'.....................................................................................
'.....................................................................................

'.....................................................................................
'.........................  Form-level COMMENTS  .....................................
'Commentary...
'.....................................................................................
'.....................................................................................



'*************************************************************************************
'*************************************************************************************
'*************************  FORM-LEVEL DECLARATIONS  *************************************
'*************************************************************************************
'*************************************************************************************
Option Explicit
Option Compare Binary '...use cAsE senSITive string comparisons



'*************************************************************************************
'*************************************************************************************
'*******************************  PRIVATE MEMBERS  ***********************************
'*************************************************************************************
'*************************************************************************************

'**************************************
'FORM-WIDE CONSTANTS
'**************************************
'Strings...
Private Const m_sFORM_NAME As String = "frmWshTypePicker"

'Bytes...
Private Const m_yLST_BOX_WSH_TYPE_NM_COL = 0
Private Const m_yLST_BOX_WSH_DEFINED_COL = 1


'**************************************
'MODULE/Form-WIDE VARIABLES
'**************************************
'Strings...
Private m_sWshTypeNmColHdg As String

'Booleans...
Private m_bUserCancel As Boolean



'=========================================================
'==================  (Private) EVENTS  ===================
'=========================================================
'=====================================================================================
Private Sub UserForm_Initialize()
'[VBA QUIRK:  Errors in Intialize, Activate, and Terminate event handlers _
```

```
cannot be handled by procedures which create or destroy instances of the _
class. If any such error-handling IS required, create a separate public _
procedure - e.g., the Initialize procedure, below.]

End Sub
'==========================================================================
Private Sub UserForm_Terminate()
'[VBA QUIRK:  Errors in Intialize, Activate, and Terminate event handlers _
cannot be handled by procedures which create or destroy instances of the _
class. If any such error-handling IS required, create a separate public _
procedure - e.g., the Initialize procedure, below.]

'Since there's really nothing useful an error-handler can do at this point...
On Error Resume Next

End Sub
'==========================================================================
Private Sub btnOK_Click()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "btnTxtOK_Click()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim sWshType As String
Dim ansCloseForm As VbMsgBoxResult

On Error GoTo ErrorHandler

'Trap for nothing picked...
If Me.lstWshTypes.ListIndex = -1 Then
    g_cMsgBoxHandler.AskCloseWshTypePickerFormEvenThoNoWshTypeSelectedQuest
    ansCloseForm = g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbYesNo)
    If ansCloseForm = vbYes Then
        Me.Hide
    Else '...i.e., No
        GoTo ExitPoint
    End If
Else '...selection made
    sWshType = _
        Trim(Me.lstWshTypes.List(Me.lstWshTypes.ListIndex, _
                m_yLST_BOX_WSH_TYPE_NM_COL))
    g_cFormsHandler.UserSelectWshTypeFromWshTypePickerForm WshTypeName:=sWshType
    If g_cDimHandler.AnyWshTypesWithUndefinedMandDimens( _
            RanOkRESULT:=bCalledProcedureRanOk) Then Me.Show
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
End If

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
m_bUserCancel = False
Exit Sub

ErrorHandler:
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'==========================================================================
Private Sub btnCancel_Click()
g_cFormsHandler.UserWantsToCxlCloseForm m_sFORM_NAME
```

```
frmWshTypePicker - 3

End Sub
'================================================================================
Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
'......................
'MSDN on...
'Cancel:  "Setting this argument to any value other than 0 [i.e., other than _
FALSE, i.e., "Setting this argument to TRUE"...] stops the QueryClose event in _
all loaded user forms and prevents the UserForm and application from closing."
'CloseMode:  "A value or constant indicating the cause of the QueryClose event"
'......................

'NOTE:  vbFormControlMenu ==> _
    The user has chosen the Close command from the Control menu on the UserForm.
'NOTE:  False === 0

'Route any X-close button calls through the btnCancel_Click procedure...
If CloseMode = vbFormControlMenu Then
    g_cMsgBoxHandler.SetPlsUseOKCxlBtnsMsg
    g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbExclamation
    Cancel = True 'i.e., -1
End If
End Sub




'*******************************************************************************
'*******************************************************************************
'*****************************  PUBLIC MEMBERS  ********************************
'*******************************************************************************
'*******************************************************************************

'============================================================
'======================  PROPERTIES  ========================
'============================================================
'================================================================================
'Property get UserCancel (read-only)...
Property Get UserCancel() As Boolean
    UserCancel = m_bUserCancel
End Property


'============================================================
'=================  (Public) PROCEDURES  ====================
'============================================================
'================================================================================
Public Sub ClearAndRepopulateListBox(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "ClearAndRepopulateListBox()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wsSrcWbkValLists As Worksheet
Dim rngWshTypesTbl As Range
Dim sWshTypeNmFmTbl As String
Dim sWshTypeDefinedFmTbl As String
Dim i As Byte

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...

On Error GoTo ErrorHandler
'Assume success until an error is encountered...
RanOkRESULT = True

If g_bStepThruMode Then Stop
Me.lstWshTypes.Clear
```

```
Set wsSrcWbkValLists = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
Set rngWshTypesTbl = wsSrcWbkValLists.Range("valtbldodynWshTypeInfo")

For i = 1 To rngWshTypesTbl.Rows.Count
    sWshTypeNmFmTbl = Trim(rngWshTypesTbl.Cells(i, g_yWSH_TYPE_TBL_NM_COL))
    sWshTypeDefinedFmTbl = rngWshTypesTbl.Cells(i, g_yWSH_TYPE_TBL_DEFINED_COL)
    'NOTE:  rows & columns in combo box are zero-based...
    Me.lstWshTypes.AddItem sWshTypeNmFmTbl
    Me.lstWshTypes.List(i - 1, g_yWSH_TYPE_TBL_DEFINED_COL - 1) = _
            sWshTypeDefinedFmTbl
Next i

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sFORM_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub

'================================================================================
'================================================================================
'================================================================================
'================================================================================
'================================================================================
'================================================================================
```

ErrorHandling - 1

```
'.....................................................................................
'.....................................................................................
'.......        "Written" by:          William H. White (consultant)
'.......        With:                  TEKsystems Inc.
'.......                               www.teksystems.com
'.......        For:                   AIG
'.......                               Financial Information Systems
'.......                               1 NY Plaza, 15th floor
'.......        Current contact:       william.white@aig.com
'.......                               (212) 551-5846
'.......        Permanent contact:     www.rcpconsulting.biz
'.......                               billwhite@rcpconsulting.biz
'.......                               New Providence, NJ
'.......        Module created:        8/8/2013
'.......        Proj finished:         11/21/2013
'.......        Proj revised:          1/15/2014
'.....................................................................................
'.....................................................................................

'.....................................................................................
'.........................  Class-level COMMENTS  .....................................
'With some adaptations, the code in this module is taken from:
    'Professional Excel Development (2nd Ed.), by:
    'Rob Bovey, Dennis Wallentin, Stephen Bullen, & John Green
    'Addison-Wesley, 2009
    'ISBN-13:  978-0-321-50879-9
    'Pgs 482-3
    'http://www.informit.com/store/product.aspx?isbn=0321508793
'.....................................................................................
'.....................................................................................


'*************************************************************************************
'*************************************************************************************
'*************************  MODULE-LEVEL DECLARATIONS  ***********************************
'*************************************************************************************
'*************************************************************************************
Option Explicit
Option Compare Binary '...use cAsE senSITive string comparisons



'*************************************************************************************
'*************************************************************************************
'*****************************  PRIVATE MEMBERS  *************************************
'*************************************************************************************
'*************************************************************************************

'**************************************
'MODULE-WIDE CONSTANTS
'**************************************
'Strings...
Private Const m_sUSER_CANCELED As String = "User canceled macro execution"
Private Const m_sSUB_FUNC_ERROR As String = _
        "Error in called Sub or invoked Function"
Private Const m_sTESTING_ERROR As String = "Testing our error-handling"
Private Const m_sMISMATCHED_PARENT_WS_ERROR As String = _
        "Header range (cell) and data column must exist on same worksheet"
Private Const m_sNO_DATA_IN_DEFINED_RNG_ERROR As String = _
        "The range we are attempting to define contains no data"
Private Const m_sSTRUCTURE_CHANGED As String = _
    "The workbook has been altered in such a way as to invalidate the VBA code"
Private Const m_sROWCOL_REF_INVALID As String = _
    "The row or column reference exceeds the size of the named range"
Private Const m_sSECOND_SINGLETON_ATTEMPT As String = _
    "You are attempting to create a second instance of a singleton object"
Private Const m_INAPP_DIMEN_SCOPE As String = _
        "You have passed an inappropriate dimension scope for the called method"
Private Const m_sFILE_ERROR_LOG As String = "Error.log"
```

```
'*******************************************************************************
'*******************************************************************************
'*****************************  PUBLIC MEMBERS  ********************************
'*******************************************************************************
'*******************************************************************************


'============================================================
'==================  (Public) PROCEDURE  ====================
'============================================================
'================================================================================
Public Sub CentralErrorHandler(ByVal ModuleName As String, _
    ByVal ProcedureName As String, _
    ByRef StepThroughErrorModeRESULT As Boolean, _
    Optional ByVal FileNm As String, _
    Optional ByVal IsEntryPoint As Boolean)
'NOTE 1:  If FileNm is not specified assume error was thrown by code _
from within this workbook.
'NOTE 2:  The boolean returned by this function sets whether or not you _
wind up stepping through the results of the error.

Static sErrMsgOrig As String
Static sFileNmOrigErr As String
Static sProcNmOrgErr As String
Dim iFile As Integer
Dim lErrNmbr As Long
Dim sErrMsgCurr As String
Dim sFullSource As String
Dim sPath As String
Dim sLogText As String
Dim sErrLogFile As String
Dim i As Integer

'Grab error info before it's cleared by On Error Resume Next below...
lErrNmbr = Err.Number

Select Case lErrNmbr
    Case g_lUSER_CANCEL
        sErrMsgCurr = m_sUSER_CANCELED
    Case g_lERR_MONKEY_WRENCH
        'We've thrown a deliberate error...
        sErrMsgCurr = m_sTESTING_ERROR
    Case g_lERR_HANDLED
        'Error in a call sub or invoked function...
        sErrMsgCurr = m_sSUB_FUNC_ERROR
    Case g_lERR_MISMATCHED_WSHS
        'Ranges must reside on same worksheet...
        sErrMsgCurr = m_sMISMATCHED_PARENT_WS_ERROR
    Case g_lERR_NO_DATA_IN_RNG
        sErrMsgCurr = m_sNO_DATA_IN_DEFINED_RNG_ERROR
    Case g_lERR_STRUCT_ALTERED
        'Some wksheet, table, range, etc. has wrong number of cols...
        sErrMsgCurr = m_sSTRUCTURE_CHANGED
    Case g_lERR_REF_OUTSIDE_RNG
        'Trying, for example, to size a 5-col range on col #6...
        sErrMsgCurr = m_sROWCOL_REF_INVALID
    Case g_lERR_2ND_SINGLETON
        'Our ObjectFactory won't allow another instance of certain objects...
        sErrMsgCurr = m_sSECOND_SINGLETON_ATTEMPT
    Case g_lERR_INAPPROP_DIMEN_SCOPE
        'Inappropriate dimension scope for called sub or funtion...
        sErrMsgCurr = m_INAPP_DIMEN_SCOPE
    Case Else
        sErrMsgCurr = Err.Description
End Select

'If this is the originating error the static error variables will be empty. _
In that case store the originating error information in these variables...
If sErrMsgOrig = vbNullString Then sErrMsgOrig = sErrMsgCurr
If sProcNmOrgErr = vbNullString Then sProcNmOrgErr = ProcedureName
```

```
ErrorHandling - 3

If sFileNmOrigErr = vbNullString Then
    If FileNm = vbNullString Then
        'When no file name is supplied assume the error rose in this file...
        sFileNmOrigErr = ThisWorkbook.name
    Else '...file name WAS supplied
        sFileNmOrigErr = FileNm
    End If
End If

'Since we can't allow errors in the central error handler itself...
On Error Resume Next '...NOTE:  This clears all properties of the Err obj

'If no file name is supplied load the default file name...
If FileNm = vbNullString Then FileNm = ThisWorkbook.name

'NOTE:
    'Chr(92) = backslash - i.e., "\"
    'Chr(91) and Chr(93) = left and right square brackets - i.e., "[", "]"
    'Chr(46) = period - i.e., "."
    'Chr(32) = space - i.e., " "

''''Get the application directory [when using on desktop apps]...
'''sPath = ThisWorkbook.Path
'''If StrComp(Right$(sPath, 1), Chr(92)) <> 0 Then sPath = sPath & Chr(92)

'Get the MyDocs directory [use this for SharePoint hosted apps]...
Dim WshShell As Shell32.Shell
Set WshShell = CreateObject("WScript.Shell")
sPath = WshShell.SpecialFolders("MyDocuments")

'Construct the fully-qualified error source name...
sFullSource = Chr(91) & FileNm & Chr(93) & ModuleName & Chr(46) & ProcedureName

'Create the error text to be logged...
sLogText = _
    Chr(32) & sFullSource & ", Error " & CStr(lErrNmbr) & ":  " & sErrMsgCurr

'Open the log file, write out the error information, and then close the file...
iFile = FreeFile()
Open sPath & ThisWorkbook.name & m_sFILE_ERROR_LOG For Append As #iFile
Print #iFile, Format$(Now(), "mm/dd/yyyy hh:mm:ss"); sLogText
If IsEntryPoint Then Print #i,
Close #iFile

'Generate messages, if and where appropriate, and do other clean-up...
Application.ScreenUpdating = True
sErrLogFile = sPath & ThisWorkbook.name & m_sFILE_ERROR_LOG
If StrComp(sErrMsgCurr, m_sUSER_CANCELED) <> 0 Then
    'Show the error message immediately if we are in debug mode.  Otherwise, _
    show the error message when we reach the entry point...
    If IsEntryPoint Or g_bDebugMode Then
        g_cMsgBoxHandler.SetFatalErrorMsg _
            ErrLogFileName:=sErrLogFile, _
            OriginatingErrFileName:=sFileNmOrigErr, _
            OriginatingProcName:=sProcNmOrgErr, _
            OriginatingErrMsg:=sErrMsgOrig
        g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbCritical
        'So that we are ready to handle the next error, clear the static _
        variables once we've reached the entry point...
        sErrMsgOrig = vbNullString
        sFileNmOrigErr = vbNullString
        sProcNmOrgErr = vbNullString
    End If
    'The return value is the debug mode status...
    StepThroughErrorModeRESULT = g_bDebugMode
Else 'This is a UserCanceled error.
    'Show the error message when we reach the entry point, but only _
    if we are in production (i.e., not in debug) mode...
    If IsEntryPoint And Not g_bDebugMode Then
        g_cMsgBoxHandler.SetUserStoppedMacroMsg ErrLogFileName:=sErrLogFile
        g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbInformation
```

```
ErrorHandling - 4

    End If
    If IsEntryPoint Then sErrMsgOrig = vbNullString
    StepThroughErrorModeRESULT = False
End If
End Sub
'================================================================================
'================================================================================
```

```
'.....................................................................
'.....................................................................
'.......        Developer:              William H. White (consultant)
'.......        With:                   TEKsystems Inc.
'.......                                www.teksystems.com
'.......        For:                    AIG
'.......                                Financial Information Systems
'.......                                1 NY Plaza, 15th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         8/8/2013
'.......        Proj finished:          11/21/2013
'.......        Proj revised:           1/15/2014
'.....................................................................
'.....................................................................

'.....................................................................
'.............  REFERENCE LIBRARIES EMPLOYED/REQUIRED BY PROJECT  .............
'''Microsoft Shell Controls and Automation
'''Microsoft Scripting Runtime
'''IF USING THE MSSSDataConnectionClass include reference to highest numbered _
'''Microsoft ActiveX Data Objects 2.X Library
'
'.....................................................................
'.....................................................................

'.....................................................................
'.........................  Class-level COMMENTS  ........................
'This module simply contains all global variable declarations
'.....................................................................
'.....................................................................


'*********************************************************************
'*********************************************************************
'************************  MODULE-LEVEL DECLARATIONS  ************************
'*********************************************************************
'*********************************************************************
Option Explicit


'=====================================================================
'=====================================================================
'************************************
'CONSTANTS
'************************************
'Longs...
'[NOTE:  numbers 1024 through 65535 are available for custom errors]
Public Const g_lERR_HANDLED As Long = 9999
Public Const g_lERR_MONKEY_WRENCH As Long = 9876
Public Const g_lERR_MISMATCHED_WSHS As Long = 9998
Public Const g_lERR_NO_DATA_IN_RNG As Long = 9997
Public Const g_lERR_STRUCT_ALTERED As Long = 9996
Public Const g_lERR_REF_OUTSIDE_RNG As Long = 9995
Public Const g_lERR_2ND_SINGLETON As Long = 9994
Public Const g_lERR_INAPPROP_DIMEN_SCOPE As Long = 9993

Public Const g_lUSER_CANCEL As Long = 18

Public Const g_lFIELD_COLOR_DISABLED As Long = -2147483633 '...&H8000000F& = ButtonFace color
Public Const g_lFIELD_COLOR_ENABLED As Long = -2147483643 '...&H80000005& = Window Backround color

'Bytes...
Public Const g_yWSH_TYPE_TBL_NM_COL As Byte = 1
Public Const g_yWSH_TYPE_TBL_DEFINED_COL As Byte = 2
Public Const g_yWSH_ASSOC_TBL_WSH_NM_COL As Byte = 1
Public Const g_yWSH_ASSOC_TBL_TYPE_COL As Byte = 2
Public Const g_yDIM_TBL_NM_COL As Byte = 1
```

```
Public Const g_yDIM_TBL_MAND_COL As Byte = 2
Public Const g_yDIM_TBL_SLRCOL_COL As Byte = 3
Public Const g_yDIM_TBL_FD_HC_COL As Byte = 4
Public Const g_yDIM_TBL_FD_RR_COL As Byte = 5
Public Const g_yDT_COL_OFF_NM_2_MAND As Byte = 1
Public Const g_yDT_COL_OFF_NM_2_SLRCOL As Byte = 2
Public Const g_yDT_COL_OFF_NM_2_FD_HC As Byte = 3
Public Const g_yDT_COL_OFF_NM_2_FD_RR As Byte = 4
Public Const g_yDT_COL_OFF_MAND_2_FD_HC As Byte = 2
Public Const g_yDT_COL_OFF_MAND_2_FD_RR As Byte = 3

Public Const g_yTTL_NO_DIMENS As Byte = 25
Public Const g_yMAND_DIMENS_EX_USD As Byte = 12
Public Const g_yCOL_WIDTH_DIMEN_LOCS As Byte = 25

Public Const g_yCOL_DIMEN_YR As Byte = 1
Public Const g_yCOL_DIMEN_PER As Byte = 2
Public Const g_yCOL_DIMEN_SCENAR As Byte = 3
Public Const g_yCOL_DIMEN_DATA_SRC As Byte = 4
Public Const g_yCOL_DIMEN_MV_ACCT As Byte = 5
Public Const g_yCOL_DIMEN_OPER_LNS As Byte = 6
Public Const g_yCOL_DIMEN_LDGR As Byte = 7
Public Const g_yCOL_DIMEN_TRANS_TYP As Byte = 8
Public Const g_yCOL_DIMEN_LGL_ENT As Byte = 9
Public Const g_yCOL_DIMEN_PARTNER As Byte = 10
Public Const g_yCOL_DIMEN_MGD_BUS As Byte = 11
Public Const g_yCOL_DIMEN_FUNC As Byte = 12
Public Const g_yCOL_DIMEN_ORIG_GEOG As Byte = 13
Public Const g_yCOL_DIMEN_PROD As Byte = 14
Public Const g_yCOL_DIMEN_CUST_TYPE As Byte = 15
Public Const g_yCOL_DIMEN_DISTRIB As Byte = 16
Public Const g_yCOL_DIMEN_RISK_LOC As Byte = 17
Public Const g_yCOL_DIMEN_PROJ As Byte = 18
Public Const g_yCOL_DIMEN_ACC_YR As Byte = 19
Public Const g_yCOL_DIMEN_UND_YR As Byte = 20
Public Const g_yCOL_DIMEN_RSVD1 As Byte = 21
Public Const g_yCOL_DIMEN_RSVD2 As Byte = 22
Public Const g_yCOL_DIMEN_TRANS_CURR As Byte = 23
Public Const g_yCOL_DIMEN_LOC_CURR As Byte = 24
Public Const g_yCOL_DIMEN_FUNC_CURR As Byte = 25
Public Const g_yCOL_DIMEN_USD_AMMT As Byte = 29


'Strings...
Public Const g_sAPP_TITLE As String = "CART Staging Layer Report"
Public Const g_sWSH_PASSWORD As String = "StrongPwd123"
Public Const g_svMsbBoxTitleInDev As String = "IN DEVELOPMENT!!!"
Public Const g_sWSH_TYPE_ALL_PC As String = "All" '...PC => Proper Case
Public Const g_sWSH_VAL_LISTS As String = "SLValdtnLists"
Public Const g_sWSH_DIMEN_LOCS As String = "SLDimenLocs"
Public Const g_sMY_TXT_BOX_PREFIX As String = "txt"
Public Const g_sMY_REFEDIT_PREFIX As String = "ref"

Public Const g_sDIM_LOC_TBL_HDR_DIM_NM As String = "Dimension"
Public Const g_sDIM_LOC_TBL_HDR_MAND As String = "Mandatory"
Public Const g_sDIM_LOC_TBL_HDR_SLR_COL As String = "SLR Column"
Public Const g_sDIM_LOC_TBL_HDR_FILE_HC As String = "File Hard Coding"
Public Const g_sDIM_LOC_TBL_HDR_FILE_RR As String = "File Range Ref"

Public Const g_sDIMEN_YR As String = "Year"
Public Const g_sDIMEN_PER As String = "Period"
Public Const g_sDIMEN_SCENAR As String = "Scenario"
Public Const g_sDIMEN_DATA_SRC As String = "Data Source"
Public Const g_sDIMEN_MV_ACCT As String = "MV Account"
Public Const g_sDIMEN_OPER_LNS As String = "Operating Lines"
Public Const g_sDIMEN_LDGR As String = "Ledger"
Public Const g_sDIMEN_TRANS_TYP As String = "Transaction Type"
Public Const g_sDIMEN_LGL_ENT As String = "Legal Entity"
Public Const g_sDIMEN_PARTNER As String = "Trading Partner"
Public Const g_sDIMEN_MGD_BUS As String = "Managed Business"
Public Const g_sDIMEN_FUNC As String = "Function"
```

```
Public Const g_sDIMEN_ORIG_GEOG As String = "Originating Geography"
Public Const g_sDIMEN_PROD As String = "Product"
Public Const g_sDIMEN_CUST_TYPE As String = "Customer Type"
Public Const g_sDIMEN_DISTRIB As String = "Distribution Channel"
Public Const g_sDIMEN_RISK_LOC As String = "Risk Location"
Public Const g_sDIMEN_PROJ As String = "Project / Initiative"
Public Const g_sDIMEN_ACC_YR As String = "Accident Year"
Public Const g_sDIMEN_UND_YR As String = "Underwriting Year"
Public Const g_sDIMEN_RSVD1 As String = "Reserved1"
Public Const g_sDIMEN_RSVD2 As String = "Reserved2"
Public Const g_sDIMEN_CURR_TRANS As String = "Transaction Currency"
Public Const g_sDIMEN_CURR_LOC As String = "Local Currency"
Public Const g_sDIMEN_CURR_FUNC As String = "Functional Currency"
Public Const g_sDIMEN_SCALE As String = "Scale"
Public Const g_sDIMEN_AMT_TYPE As String = "Amount Type"

Public Const g_sSCALE_SEE_WSH As String = "Varies by worksheet"
Public Const g_sSCALE_NONE As String = "None"
Public Const g_sSCALE_THOUS As String = "Thousands"
Public Const g_sSCALE_MILLS As String = "Millions"
Public Const g_sSCALE_BILLS As String = "Billions"
Public Const g_ySCALE_OPTIONS As Byte = 5

'Booleans...
Public g_bUserRenamedOrDeletedDataWshs As Boolean

'**************************************
'VARIABLES
'**************************************
'Classes...
Public g_cMsgBoxHandler As MsgBoxHandler
Public g_cObjFactory As ObjectFactory
Public g_cFormsHandler As FormsHandler
Public g_cDimHandler As DimensionHandler

'Workbook...
Public g_wbSrcData As Workbook

'Strings...
Public g_sSrcWbkNm As String
Public g_sSrcWbkFullNm As String
Public g_sSrcWbkPath As String

'Booleans (default = FALSE)...
Public g_bStepThruMode As Boolean
Public g_bDebugMode As Boolean
Public g_bMandDimensOnly As Boolean
Public g_bWizardMode As Boolean




'**************************************
'ENUMERATIONS
'**************************************
Public Enum ProjDimenReq
    projDimenMandatory = 10
    projDimenOptional = 20
    projDimenReserved = 30
End Enum

Public Enum ProjDimenScope
    projScopeFile = 10
    projScopeWshType = 15
    projScopeWsh = 20
    projScopeCol = 30
    projScopeRow = 40
End Enum

Public Enum ProjAmtScale
    projScaleNoSelection = 0
```

```
    projScaleSeeWshType = 10
    projScaleNone = 20
    projScaleThous = 30
    projScaleMills = 40
    projScaleBills = 50
End Enum

Public Enum ProjAmtType
    projAmtUSD = 10
    projAmtTrans = 20
    projAmtLocal = 30
    projAmtFunc = 40
End Enum

Public Enum ProjMissingWsh
    projMissingValLists = 10
    projMissingDimenLocs = 20
End Enum

Public Enum ProjAssocDisassocWsh
    projAssoc = 10
    projDisassoc = 20
End Enum

Public Enum ProjCtrlType
    projCtrlTypTxtBox = 10
    projCtrlTypeRefEdit = 20
End Enum

Public Enum ProjNextStage
    projNxtStgOpenSrc = 10
    projNxtStgFileDimens = 20
    projNxtStgIDDataWshs = 30
    projNxtStgCreateWshTyps = 40
    projNxtStgAssocWshTyps = 50
    projNxtStgWshTypeDimens = 55
    projNxtStgWshDimens = 60
    projNxtStgRowDimens = 70
    projNxtStgColDimens = 80
    projNxtStgImportData = 90
    projNxtStgExportToCSV = 100
End Enum
```

```
RibbonHandler - 1

'.....................................................................................
'.....................................................................................
'.......        Developer:              William H. White (consultant)
'.......        With:                   TEKsystems Inc.
'.......                                www.teksystems.com
'.......        For:                    AIG
'.......                                Financial Information Systems
'.......                                1 NY Plaza, 15th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         8/8/2013
'.......        Proj finished:          11/21/2013
'.......        Proj revised:           1/15/2014
'.....................................................................................
'.....................................................................................

'.....................................................................................
'.........................  Module-level COMMENTS  ...........................
'Commentary...
'.....................................................................................
'.....................................................................................


'*************************************************************************************
'*************************************************************************************
'************************  MODULE-LEVEL DECLARATIONS  ************************
'*************************************************************************************
'*************************************************************************************
Option Explicit
Option Compare Binary '...use cAsE senSITive string comparisons




'*************************************************************************************
'*************************************************************************************
'*******************************  PRIVATE MEMBERS  *******************************
'*************************************************************************************
'*************************************************************************************

'**************************************
'MODULE-WIDE CONSTANTS
'**************************************
'Strings...
Private Const m_sMODULE_NAME As String = "RibbonHandler"
Private Const m_sVERSION As String = "2014.01.15"


'**************************************
'MODULE-WIDE VARIABLES
'**************************************
'Classes...
Private m_FilePrep As FileHandler

'Enums...
Private m_enumNextStage As ProjNextStage

'Ribbon...
Private m_Ribbon As IRibbonUI

'Booleans...
Private m_bOnly1WshType As Boolean
Private m_bEnableAssocWshTypesBtnSpecCase As Boolean



'=========================================================
'==================  (Private) PROCEDURES  =================
```

```
RibbonHandler - 2

'============================================================
'============================================================================
Private Sub DarnedFoolUserClosedSourceWbk(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "DarnedFoolUserClosedSourceWbk()"
Dim bCalledProcedureRanOk As Boolean
'''Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'OPERATING CODE HERE...
g_cMsgBoxHandler.SetSourceFileClosedMsg
g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbCritical
m_FilePrep.HandleUserClosingSourceWbk _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'''Must-run procedure/function clean-up code...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'============================================================================
Public Sub SetDimensionsMaestro( _
    ByVal DimScope As ProjDimenScope, _
    ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "SetDimensionsMaestro()"
Dim bCalledProcedureRanOk As Boolean
'''Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'OPERATING CODE HERE...
'Trap for user having closed the workbook...
If Not m_FilePrep.IsSourceWbkStillOpen( _
        RanOkRESULT:=bCalledProcedureRanOk) Then
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    DarnedFoolUserClosedSourceWbk RanOkRESULT:=bCalledProcedureRanOk
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    GoTo ExitPoint
Else
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
End If

'If user had previously opened and then canceled the File Dimension _
form we set g_cDimHandler to nothing.  Ergo, in case the user _
IS reopening this form after a prior open-and-cancel.  (The procedure, btw, _
does nothing if the object already exists)...
```

```
RibbonHandler - 3

g_cObjFactory.CreateGlobalDimensionHandler _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

If g_bStepThruMode Then Stop
g_wbSrcData.Activate
Select Case DimScope
    Case projScopeFile
        g_cFormsHandler.OpenWbkDimenIDForm RanOkRESULT:=bCalledProcedureRanOk
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    Case projScopeWshType
        If m_bOnly1WshType Then
            If g_cFormsHandler.WshTypeDimensFormsColl.Count = 1 Then
                g_cFormsHandler.WshTypeDimensFormsColl.Item(1).Show
            Else
                g_cFormsHandler.OpenWshTypeDimenIDForm _
                        OpenedFmAnotherWTDForm:=False, _
                        RanOkRESULT:=bCalledProcedureRanOk, _
                        WshTypeNm:=g_sWSH_TYPE_ALL_PC
                    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
            End If
        Else
            g_cFormsHandler.OpenAndPopulateWshTypePickerForm _
                    RanOkRESULT:=bCalledProcedureRanOk
                If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        End If
    Case Else
        Err.Raise g_lERR_INAPPROP_DIMEN_SCOPE
End Select

ExitPoint:
On Error Resume Next
'''Must-run procedure/function clean-up code...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'=============================================================================
Private Sub CreateWshTypesMaestro(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "CreateWshTypesMaestro()"
Dim bCalledProcedureRanOk As Boolean
'''Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'OPERATING CODE HERE...
'Trap for user having closed the workbook...
If Not m_FilePrep.IsSourceWbkStillOpen( _
        RanOkRESULT:=bCalledProcedureRanOk) Then
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    DarnedFoolUserClosedSourceWbk RanOkRESULT:=bCalledProcedureRanOk
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
Else '...wbk IS still open
```

```
RibbonHandler - 4

    g_wbSrcData.Activate
    'If g_bStepThruMode Then Stop
    g_cFormsHandler.OpenAndPopulateAddRemWshTypeForm _
            RanOkRESULT:=bCalledProcedureRanOk
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    If Not g_bWizardMode Then ThisWorkbook.Activate
End If

ExitPoint:
On Error Resume Next
'''Must-run procedure/function clean-up code...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Private Sub AssociateWshsWithWshTypesMaestro(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "AssociateWshsWithWshTypesMaestro()"
Dim bCalledProcedureRanOk As Boolean
'''Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'OPERATING CODE HERE...
'If g_bStepThruMode Then Stop
'Trap for user having closed the workbook...
If Not m_FilePrep.IsSourceWbkStillOpen( _
        RanOkRESULT:=bCalledProcedureRanOk) Then
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    DarnedFoolUserClosedSourceWbk RanOkRESULT:=bCalledProcedureRanOk
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
Else '...wbk IS still open
    'If g_bStepThruMode Then Stop
    g_wbSrcData.Activate
    'If g_bStepThruMode Then Stop
    g_cFormsHandler.OpenAndPopulateAssocWshTypeWithWshForm _
            RanOkRESULT:=bCalledProcedureRanOk
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
End If
If Not g_bWizardMode Then ThisWorkbook.Activate

ExitPoint:
On Error Resume Next
'''Must-run procedure/function clean-up code...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
```

```
RibbonHandler - 5

                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'============================================================================
Private Sub SetModuleBooleanFor1WshType(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "SetModuleBooleanFor1WshType()"
'''Operating code declarations...
Dim wsValLists As Worksheet
Dim rngWshTypesTbl As Range

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'OPERATING CODE HERE...
Set wsValLists = _
        g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
Set rngWshTypesTbl = wsValLists.Range("vallstdodynWshTypes")
If rngWshTypesTbl.Rows.Count = 1 And _
        Not IsEmpty(rngWshTypesTbl.Cells(1, g_yWSH_TYPE_TBL_NM_COL)) Then _
        m_bOnly1WshType = True

ExitPoint:
On Error Resume Next
'''Must-run procedure/function clean-up code...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub




'*****************************************************************************
'*****************************************************************************
'*****************************  PUBLIC MEMBERS  ******************************
'*****************************************************************************
'*****************************************************************************


'============================================================
'=================  (Public) PROCEDURES  ==================
'============================================================
'============================================================================
Public Sub rxCustomUI_onLoad(rb As IRibbonUI)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "rxcustomUI_onLoad()"
```

```
On Error GoTo ErrorHandler

'Instantiate module-level variables...
Set m_Ribbon = rb
Set m_FilePrep = New FileHandler

ExitPoint:
On Error Resume Next
Exit Sub

ErrorHandler:
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
            ModuleName:=m_sMODULE_NAME, _
            ProcedureName:=sSOURCE, _
            StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
            If g_bDebugMode Then Stop
            'So we can step through the code which caused the error...
            Resume
        Else
            Resume ExitPoint
        End If
End Sub
'============================================================================
Public Sub btns_Click(control As IRibbonControl)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "btns_Click()"
Dim bCalledProcedureRanOk As Boolean
'''Operating code declarations...
Dim frmWTD As frmWshTypeDimens
Dim sWshType As String

On Error GoTo ErrorHandler
'NOTE:  In WizardMode there can be any number of forms in memory when the _
call stack unwinds back to here.  We want to destroy those open forms.  The _
"original" form will "always" need to be destroyed, though we still need to _
test for its existence.  If the user canceled out of the form our Cancel/Close _
procedure destroys the form...


'If g_bStepThruMode Then Stop
'Determine which button was pressed & invoke appropriate method...
Select Case control.ID
    Case "btnOpenSourceFile"
        'If g_bStepThruMode Then Stop
        m_FilePrep.OpenSourceWbkMaestro RanOkRESULT:=bCalledProcedureRanOk
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
     Case "btnSetFileDimens"
        If g_bStepThruMode Then Stop
        SetDimensionsMaestro _
                DimScope:=projScopeFile, _
                RanOkRESULT:=bCalledProcedureRanOk
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        'Once the call stack unwinds back to here we want to destroy _
        the open forms...
        If g_bWizardMode Then
            'Close any WshTypeDimen forms...
            If g_cFormsHandler.WshTypeDimensFormsColl.Count > 1 Then
                For Each frmWTD In g_cFormsHandler.WshTypeDimensFormsColl
                    sWshType = frmWTD.WshTypeNm
                    g_cFormsHandler.UnloadForm _
                            FormName:=frmWTD.name, _
                            RanOkRESULT:=bCalledProcedureRanOk, _
                            WshTypeNm:=sWshType
                       If Not bCalledProcedureRanOk Then _
                                Err.Raise g_lERR_HANDLED
                Next frmWTD
```

```
        End If
        'Close the WshTypePicker form...
        If g_cFormsHandler.IsFormOpen( _
                FormName:=frmWshTypePicker.name, _
                RanOkRESULT:=bCalledProcedureRanOk) Then
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
            g_cFormsHandler.UnloadForm _
                    FormName:=frmWshTypePicker.name, _
                    RanOkRESULT:=bCalledProcedureRanOk
                If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        Else
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        End If
        'Close the WshTypeWshNameAssociation form...
        If g_cFormsHandler.IsFormOpen( _
                FormName:=frmTypeWshAssoc.name, _
                RanOkRESULT:=bCalledProcedureRanOk) Then
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
            g_cFormsHandler.UnloadForm _
                    FormName:=frmTypeWshAssoc.name, _
                    RanOkRESULT:=bCalledProcedureRanOk
                If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        Else
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        End If
        'Close the Add/Remove WshTypes form...
        If g_cFormsHandler.IsFormOpen( _
                FormName:=frmAddRemWshTypes.name, _
                RanOkRESULT:=bCalledProcedureRanOk) Then
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
            g_cFormsHandler.UnloadForm _
                    FormName:=frmAddRemWshTypes.name, _
                    RanOkRESULT:=bCalledProcedureRanOk
                If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        Else
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        End If
        'Close the Data Worksheets to Import form...
        If g_cFormsHandler.IsFormOpen( _
                FormName:=frmWshsToImport.name, _
                RanOkRESULT:=bCalledProcedureRanOk) Then
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
            g_cFormsHandler.UnloadForm _
                    FormName:=frmWshsToImport.name, _
                    RanOkRESULT:=bCalledProcedureRanOk
                If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        Else
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        End If
    End If
    If g_cFormsHandler.IsFormOpen( _
            FormName:=frmWbkMandDimens.name, _
            RanOkRESULT:=bCalledProcedureRanOk) Then
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        g_cFormsHandler.UnloadForm _
                FormName:=frmWbkMandDimens.name, _
                RanOkRESULT:=bCalledProcedureRanOk
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    Else
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    End If
Case "btnIdDataWshs"
    'If g_bStepThruMode Then Stop
    g_cFormsHandler.OpenAndPopulateDataWshsForm _
            RanOkRESULT:=bCalledProcedureRanOk
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    'Once the call stack unwinds back to here we want to destroy _
    the open forms...
    'If g_bStepThruMode Then Stop
    If g_bWizardMode Then
        'Close any WshTypeDimen forms...
```

```
        If g_cFormsHandler.WshTypeDimensFormsColl.Count > 1 Then
            For Each frmWTD In g_cFormsHandler.WshTypeDimensFormsColl
                sWshType = frmWTD.WshTypeNm
                g_cFormsHandler.UnloadForm _
                        FormName:=frmWTD.name, _
                        RanOkRESULT:=bCalledProcedureRanOk, _
                        WshTypeNm:=sWshType
                    If Not bCalledProcedureRanOk Then _
                         Err.Raise g_lERR_HANDLED
            Next frmWTD
        End If
        'Close the WshTypePicker form...
        If g_cFormsHandler.IsFormOpen( _
                FormName:=frmWshTypePicker.name, _
                RanOkRESULT:=bCalledProcedureRanOk) Then
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
            g_cFormsHandler.UnloadForm _
                    FormName:=frmWshTypePicker.name, _
                    RanOkRESULT:=bCalledProcedureRanOk
                If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        Else
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        End If
        'Close the WshTypeWshNameAssociation form...
        If g_cFormsHandler.IsFormOpen( _
                FormName:=frmTypeWshAssoc.name, _
                RanOkRESULT:=bCalledProcedureRanOk) Then
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
            g_cFormsHandler.UnloadForm _
                    FormName:=frmTypeWshAssoc.name, _
                    RanOkRESULT:=bCalledProcedureRanOk
                If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        Else
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        End If
        'Close the Add/Remove WshTypes form...
        If g_cFormsHandler.IsFormOpen( _
                FormName:=frmAddRemWshTypes.name, _
                RanOkRESULT:=bCalledProcedureRanOk) Then
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
            g_cFormsHandler.UnloadForm _
                    FormName:=frmAddRemWshTypes.name, _
                    RanOkRESULT:=bCalledProcedureRanOk
                If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        Else
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        End If
    End If
    'This will always need to get unloaded...
    If g_cFormsHandler.IsFormOpen( _
            FormName:=frmWshsToImport.name, _
            RanOkRESULT:=bCalledProcedureRanOk) Then
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        g_cFormsHandler.UnloadForm _
                FormName:=frmWshsToImport.name, _
                RanOkRESULT:=bCalledProcedureRanOk
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    Else
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    End If

Case "btnAddRemTypes"
    If g_bStepThruMode Then Stop
    CreateWshTypesMaestro RanOkRESULT:=bCalledProcedureRanOk
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    'Once the call stack unwinds back to here we want to destroy _
    the open forms...
    'If g_bStepThruMode Then Stop
    If g_bWizardMode Then
        'Close any WshTypeDimen forms...
        If g_cFormsHandler.WshTypeDimensFormsColl.Count > 1 Then
```

```
            For Each frmWTD In g_cFormsHandler.WshTypeDimensFormsColl
                sWshType = frmWTD.WshTypeNm
                g_cFormsHandler.UnloadForm _
                        FormName:=frmWTD.name, _
                        RanOkRESULT:=bCalledProcedureRanOk, _
                        WshTypeNm:=sWshType
                    If Not bCalledProcedureRanOk Then _
                            Err.Raise g_lERR_HANDLED
            Next frmWTD
        End If
        'Close the WshTypePicker form...
        If g_cFormsHandler.IsFormOpen( _
                FormName:=frmWshTypePicker.name, _
                RanOkRESULT:=bCalledProcedureRanOk) Then
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
            g_cFormsHandler.UnloadForm _
                    FormName:=frmWshTypePicker.name, _
                    RanOkRESULT:=bCalledProcedureRanOk
                If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        Else
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        End If
        'Close the WshTypeWshNameAssociation form...
        If g_cFormsHandler.IsFormOpen( _
                FormName:=frmTypeWshAssoc.name, _
                RanOkRESULT:=bCalledProcedureRanOk) Then
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
            g_cFormsHandler.UnloadForm _
                    FormName:=frmTypeWshAssoc.name, _
                    RanOkRESULT:=bCalledProcedureRanOk
                If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        Else
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        End If
    End If
    'This will always need to get unloaded...
    If g_cFormsHandler.IsFormOpen( _
            FormName:=frmAddRemWshTypes.name, _
            RanOkRESULT:=bCalledProcedureRanOk) Then
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        g_cFormsHandler.UnloadForm _
                FormName:=frmAddRemWshTypes.name, _
                RanOkRESULT:=bCalledProcedureRanOk
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    Else
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    End If

Case "btnAssocTypes"
    If g_bStepThruMode Then Stop
    AssociateWshsWithWshTypesMaestro RanOkRESULT:=bCalledProcedureRanOk
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    'Once the call stack unwinds back to here we want to destroy _
    the open forms...
    'If g_bStepThruMode Then Stop
    If g_bWizardMode Then
        'Close any WshTypeDimen forms...
        If g_cFormsHandler.WshTypeDimensFormsColl.Count > 1 Then
            For Each frmWTD In g_cFormsHandler.WshTypeDimensFormsColl
                sWshType = frmWTD.WshTypeNm
                g_cFormsHandler.UnloadForm _
                        FormName:=frmWTD.name, _
                        RanOkRESULT:=bCalledProcedureRanOk, _
                        WshTypeNm:=sWshType
                    If Not bCalledProcedureRanOk Then _
                            Err.Raise g_lERR_HANDLED
            Next frmWTD
        End If
        'Close the WshTypePicker form...
        If g_cFormsHandler.IsFormOpen( _
                FormName:=frmWshTypePicker.name, _
```

```
                RanOkRESULT:=bCalledProcedureRanOk) Then
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
            g_cFormsHandler.UnloadForm _
                    FormName:=frmWshTypePicker.name, _
                    RanOkRESULT:=bCalledProcedureRanOk
                If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        Else
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        End If
    End If
    'This will always need to get unloaded...
    If g_cFormsHandler.IsFormOpen( _
            FormName:=frmTypeWshAssoc.name, _
            RanOkRESULT:=bCalledProcedureRanOk) Then
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        g_cFormsHandler.UnloadForm _
                FormName:=frmTypeWshAssoc.name, _
                RanOkRESULT:=bCalledProcedureRanOk
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    Else
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    End If

Case "btnSetWshTypeDimens"
    If g_bStepThruMode Then Stop
    If Not m_bOnly1WshType Then
        g_cFormsHandler.OpenAndPopulateWshTypePickerForm _
                RanOkRESULT:=bCalledProcedureRanOk
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    Else
        SetDimensionsMaestro _
                DimScope:=projScopeWshType, _
                RanOkRESULT:=bCalledProcedureRanOk
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    End If
    'Once the call stack unwinds back to here we want to destroy _
    the open forms...
    'If g_bStepThruMode Then Stop
    'Close any WshTypeDimen forms...
    If g_cFormsHandler.WshTypeDimensFormsColl.Count > 1 Then
        For Each frmWTD In g_cFormsHandler.WshTypeDimensFormsColl
            sWshType = frmWTD.WshTypeNm
            g_cFormsHandler.UnloadForm _
                    FormName:=frmWTD.name, _
                    RanOkRESULT:=bCalledProcedureRanOk, _
                    WshTypeNm:=sWshType
                If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        Next frmWTD
    End If
    'Close the WshTypePicker form...
    If g_cFormsHandler.IsFormOpen( _
            FormName:=frmWshTypePicker.name, _
            RanOkRESULT:=bCalledProcedureRanOk) Then
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        g_cFormsHandler.UnloadForm _
                FormName:=frmWshTypePicker.name, _
                RanOkRESULT:=bCalledProcedureRanOk
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    Else
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    End If

Case "btnRunDataImport"
    If g_bStepThruMode Then Stop
    m_FilePrep.ImportDataMaestro RanOkRESULT:=bCalledProcedureRanOk
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Case "btnExportToCSV"
    If g_bStepThruMode Then Stop
    m_FilePrep.CreateCSVFile RanOkRESULT:=bCalledProcedureRanOk
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
```

```
RibbonHandler - 11

End Select

If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'''Must-run procedure/function clean-up code...
Exit Sub

ErrorHandler:
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Public Sub SetNextStage(ByVal NextStage As ProjNextStage)
m_enumNextStage = NextStage
End Sub
'================================================================================
Public Sub ResetRibbon()
m_Ribbon.Invalidate
End Sub
'================================================================================
Public Sub SetEnableAssocWshTypesSpecCaseFlag(ByVal BoolValue As Boolean)
m_bEnableAssocWshTypesBtnSpecCase = BoolValue
End Sub
'================================================================================
Public Sub SetNextStageAndResetRibbon(ByVal NextStage As ProjNextStage)
If g_bStepThruMode Then Stop
m_enumNextStage = NextStage
m_Ribbon.Invalidate
End Sub
'================================================================================
Public Sub SetOnly1WshTypeFlag(ByVal Only1WshType As Boolean)
m_bOnly1WshType = Only1WshType
End Sub
'================================================================================
Public Sub btns_GetEnabled(control As IRibbonControl, ByRef enabled)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "btns_GetEnabled()"
'''Operating code declarations...
Dim bOneWshType As Boolean

On Error GoTo ErrorHandler

'CODE HERE...
bOneWshType = m_bOnly1WshType

Select Case m_enumNextStage
    Case projNxtStgOpenSrc
        If StrComp(control.ID, "btnOpenSourceFile") = 0 Then enabled = True
        If StrComp(control.ID, "btnSetFileDimens") = 0 Then enabled = False
        If StrComp(control.ID, "btnIdDataWshs") = 0 Then enabled = False
        If StrComp(control.ID, "btnAddRemTypes") = 0 Then enabled = False
        'If StrComp(control.ID, "btnAssocTypes") = 0 Then enabled = False
        If StrComp(control.ID, "btnSetWshTypeDimens") = 0 Then enabled = False
        If StrComp(control.ID, "btnRunDataImport") = 0 Then enabled = False
        If StrComp(control.ID, "btnExportToCSV") = 0 Then enabled = False
    Case projNxtStgFileDimens
        If StrComp(control.ID, "btnOpenSourceFile") = 0 Then enabled = False
```

```
        If StrComp(control.ID, "btnSetFileDimens") = 0 Then enabled = True
        If StrComp(control.ID, "btnIdDataWshs") = 0 Then enabled = False
        If StrComp(control.ID, "btnAddRemTypes") = 0 Then enabled = False
        'If StrComp(control.ID, "btnAssocTypes") = 0 Then enabled = False
        If StrComp(control.ID, "btnSetWshTypeDimens") = 0 Then enabled = False
        If StrComp(control.ID, "btnRunDataImport") = 0 Then enabled = False
        If StrComp(control.ID, "btnExportToCSV") = 0 Then enabled = False
    Case projNxtStgIDDataWshs
        'I handle this enum, though we should never see it passed in...
        If StrComp(control.ID, "btnOpenSourceFile") = 0 Then enabled = False
        If StrComp(control.ID, "btnSetFileDimens") = 0 Then enabled = True
        If StrComp(control.ID, "btnIdDataWshs") = 0 Then enabled = True
        If StrComp(control.ID, "btnAddRemTypes") = 0 Then enabled = False
        'If StrComp(control.ID, "btnAssocTypes") = 0 Then enabled = False
        If StrComp(control.ID, "btnSetWshTypeDimens") = 0 Then enabled = False
        If StrComp(control.ID, "btnRunDataImport") = 0 Then enabled = False
        If StrComp(control.ID, "btnExportToCSV") = 0 Then enabled = False
    Case projNxtStgCreateWshTyps
        If StrComp(control.ID, "btnOpenSourceFile") = 0 Then enabled = False
        If StrComp(control.ID, "btnSetFileDimens") = 0 Then enabled = True
        If StrComp(control.ID, "btnIdDataWshs") = 0 Then enabled = True
        If StrComp(control.ID, "btnAddRemTypes") = 0 Then enabled = True
        'If StrComp(control.ID, "btnAssocTypes") = 0 Then enabled = False
        If StrComp(control.ID, "btnSetWshTypeDimens") = 0 Then enabled = False
        If StrComp(control.ID, "btnRunDataImport") = 0 Then enabled = False
        If StrComp(control.ID, "btnExportToCSV") = 0 Then enabled = False
    Case projNxtStgAssocWshTyps
        If StrComp(control.ID, "btnOpenSourceFile") = 0 Then enabled = False
        If StrComp(control.ID, "btnSetFileDimens") = 0 Then enabled = True
        If StrComp(control.ID, "btnIdDataWshs") = 0 Then enabled = True
        If StrComp(control.ID, "btnAddRemTypes") = 0 Then enabled = True
        'If StrComp(control.ID, "btnAssocTypes") = 0 Then enabled = True
        If StrComp(control.ID, "btnSetWshTypeDimens") = 0 Then enabled = False
        If StrComp(control.ID, "btnRunDataImport") = 0 Then enabled = False
        If StrComp(control.ID, "btnExportToCSV") = 0 Then enabled = False
    Case projNxtStgWshTypeDimens
        If StrComp(control.ID, "btnOpenSourceFile") = 0 Then enabled = False
        If StrComp(control.ID, "btnSetFileDimens") = 0 Then enabled = True
        If StrComp(control.ID, "btnIdDataWshs") = 0 Then enabled = True
        If StrComp(control.ID, "btnAddRemTypes") = 0 Then enabled = True
        If StrComp(control.ID, "btnSetWshTypeDimens") = 0 Then enabled = True
        If StrComp(control.ID, "btnRunDataImport") = 0 Then enabled = False
        If StrComp(control.ID, "btnExportToCSV") = 0 Then enabled = False
'       If bOneWshType Then
'           If StrComp(control.ID, "btnAssocTypes") = 0 Then enabled = False
'       Else
'           If StrComp(control.ID, "btnAssocTypes") = 0 Then enabled = True
'       End If
    Case projNxtStgImportData
        If StrComp(control.ID, "btnOpenSourceFile") = 0 Then enabled = False
        If StrComp(control.ID, "btnSetFileDimens") = 0 Then enabled = True
        If StrComp(control.ID, "btnIdDataWshs") = 0 Then enabled = True
        If StrComp(control.ID, "btnAddRemTypes") = 0 Then enabled = True
        If StrComp(control.ID, "btnSetWshTypeDimens") = 0 Then enabled = True
        If StrComp(control.ID, "btnRunDataImport") = 0 Then enabled = True
        If StrComp(control.ID, "btnExportToCSV") = 0 Then enabled = False
'       If bOneWshType Then
'           If StrComp(control.ID, "btnAssocTypes") = 0 Then enabled = False
'       Else
'           If StrComp(control.ID, "btnAssocTypes") = 0 Then enabled = True
'       End If
    Case projNxtStgExportToCSV
        If StrComp(control.ID, "btnOpenSourceFile") = 0 Then enabled = False
        If StrComp(control.ID, "btnSetFileDimens") = 0 Then enabled = True
        If StrComp(control.ID, "btnIdDataWshs") = 0 Then enabled = True
        If StrComp(control.ID, "btnAddRemTypes") = 0 Then enabled = True
        If StrComp(control.ID, "btnSetWshTypeDimens") = 0 Then enabled = True
        If StrComp(control.ID, "btnRunDataImport") = 0 Then enabled = True
        If StrComp(control.ID, "btnExportToCSV") = 0 Then enabled = True
'       If bOneWshType Then
'           If StrComp(control.ID, "btnAssocTypes") = 0 Then enabled = False
```

```
RibbonHandler - 13

'        Else
'            If StrComp(control.ID, "btnAssocTypes") = 0 Then enabled = True
'        End If
End Select


ExitPoint:
On Error Resume Next
'''Must-run procedure/function clean-up code...
Exit Sub

ErrorHandler:
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Public Sub btnAssocTypes_GetEnabled(control As IRibbonControl, ByRef enabled)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "btnAssocTypes_GetEnabled()"
'''Operating code declarations...
Dim bOneWshType As Boolean

On Error GoTo ErrorHandler

'CODE HERE...
If m_bOnly1WshType And Not m_bEnableAssocWshTypesBtnSpecCase Then
    enabled = False
Else
    Select Case m_enumNextStage
        Case projNxtStgAssocWshTyps
            enabled = True
        Case projNxtStgWshTypeDimens
            enabled = True
        Case projNxtStgImportData
            enabled = True
        Case projNxtStgExportToCSV
            enabled = True
        Case Else
            enabled = False
    End Select
End If

ExitPoint:
On Error Resume Next
'''Must-run procedure/function clean-up code...
Exit Sub

ErrorHandler:
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
```

```
                Resume ExitPoint
        End If
End Sub
'================================================================================
Public Sub lbls_GetLabel(control As IRibbonControl, ByRef label)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "lbls_GetCaption()"
Dim bCalledProcedureRanOk As Boolean
On Error GoTo ErrorHandler

Select Case control.ID
    Case "lblVerzNo"
        label = m_sVERSION
    Case Else
        'No other labels at this point...
End Select

ExitPoint:
On Error Resume Next
'''Must-run procedure/function clean-up code...
Exit Sub

ErrorHandler:
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
'================================================================================
'================================================================================
'================================================================================
```

```
'.....................................................................
'.....................................................................
'.......        Developer:              William H. White (consultant)
'.......        With:                   TEKsystems Inc.
'.......                                www.teksystems.com
'.......        For:                    AIG
'.......                                Financial Information Systems
'.......                                1 NY Plaza, 15th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         8/8/2013
'.......        Proj finished:          11/21/2013
'.......        Proj revised:           1/15/2014
'.....................................................................
'.....................................................................

'.....................................................................
'.........................  Module-level COMMENTS  ....................
'The procedures in this module are designed to be general-purpose - i.e., _
they are not specific to this project.
'These procedures DO, however, require integration with centralized _
error-trapping.
'.....................................................................
'.....................................................................


'*********************************************************************
'*********************************************************************
'************************  MODULE-LEVEL DECLARATIONS  ********************
'*********************************************************************
'*********************************************************************
Option Explicit
Option Compare Binary '...use cAsE senSITive string comparisons



'*********************************************************************
'*********************************************************************
'*******************************  PRIVATE MEMBERS  *******************************
'*********************************************************************
'*********************************************************************

'**************************************
'MODULE-WIDE CONSTANTS
'**************************************
'Strings...
Private Const m_sMODULE_NAME As String = "Utilities"



'*********************************************************************
'*********************************************************************
'*******************************  PUBLIC MEMBERS  *******************************
'*********************************************************************
'*********************************************************************

'============================================================
'==================  (Public) FUNCTIONS  ====================
'============================================================
'=========================================================================
Public Function DoesFolderOrFileExist(ByVal PathOrFileFullNm As String) _
    As Boolean
'*********
'NOTE:  If passing in a folder it does not matter whether or not you _
include the trailing backslash.
'*********
Dim attrPathOrFile As VbFileAttribute
```

```
'Ignore errors to allow for error evaluation...
On Error Resume Next

'We don't care about attrPathOrFile - we just want to see if we get an error...
attrPathOrFile = GetAttr(PathOrFileFullNm)

If Err.Number = 0 Then
    DoesFolderOrFileExist = True
Else
    DoesFolderOrFileExist = False
End If
End Function
'============================================================================
Public Function IsWorkbookOpen(ByVal WbName As String) As Boolean
Dim wb As Workbook
On Error Resume Next
Set wb = Application.Workbooks(WbName)
IsWorkbookOpen = Not wb Is Nothing
End Function

'============================================================
'==================  (Public) PROCEDURES  ==================
'============================================================
'============================================================================
'============================================================================
Public Sub BubbleSortArray(arr, ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "BubbleSortArray()"
'''Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim strTemp As String
Dim i As Long
Dim j As Long
Dim lngMin As Long
Dim lngMax As Long

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
lngMin = LBound(arr)
lngMax = UBound(arr)
For i = lngMin To lngMax - 1
    For j = i + 1 To lngMax
        If arr(i) > arr(j) Then
            strTemp = arr(i)
            arr(i) = arr(j)
            arr(j) = strTemp
        End If
    Next j
Next i

ExitPoint:
On Error Resume Next
'''Must-run procedure/function clean-up code...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
```

```
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'============================================================================
Public Function RepeatString( _
        ByVal StringToRepeat As String, _
        ByVal NoRepetitions As Integer, _
        ByRef RanOkRESULT As Boolean) As String
'Error-handling declarations...
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "RepeatString()"
'''Operating code declarations...
Dim sFinal As String
Dim i As Integer

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
For i = 1 To NoRepetitions
    sFinal = sFinal & StringToRepeat
Next i

RepeatString = sFinal

ExitPoint:
On Error Resume Next
'''Must-run procedure/function clean-up code...
Exit Function

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bFUNCTION_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Function
'============================================================================
'============================================================================
'============================================================================
'============================================================================
'============================================================================
'============================================================================
'============================================================================
```

```
'.....................................................................................
'.....................................................................................
'.......        Developer:              William H. White (consultant)
'.......        With:                   TEKsystems Inc.
'.......                                www.teksystems.com
'.......        For:                    AIG
'.......                                Financial Information Systems
'.......                                1 NY Plaza, 15th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         8/8/2013
'.......        Proj finished:          11/21/2013
'.......        Proj revised:           1/15/2014
'.....................................................................................
'.....................................................................................

'.....................................................................................
'.........................  Module-level COMMENTS  ...................................
'DEVELOPMENT-ONLY PROCEDURES!!!!
'Note:  Because of the nature of this module there is very little, if any, _
error-handling within this module's procedures and functions.
'.....................................................................................
'.....................................................................................


'*************************************************************************************
'*************************************************************************************
'************************  MODULE-LEVEL DECLARATIONS  ********************************
'*************************************************************************************
'*************************************************************************************
Option Explicit
Option Compare Binary '...use cAsE senSITive string comparisons



'*************************************************************************************
'*************************************************************************************
'******************************  PRIVATE MEMBERS  ************************************
'*************************************************************************************
'*************************************************************************************

'**************************************
'MODULE-WIDE CONSTANTS
'**************************************
'Strings...
Private Const m_sMODULE_NAME As String = "vDevUtilities"



'*************************************************************************************
'*************************************************************************************
'******************************  PUBLIC MEMBERS  *************************************
'*************************************************************************************
'*************************************************************************************

'============================================================
'==================  (Public) PROCEDURES  ==================
'============================================================
'=====================================================================================
Public Sub PrintXlDefinedNamesToImmediateWindow()
Dim wbNmz As Names
Dim wbDefNm As name
Dim sNameNm As String
Set wbNmz = ThisWorkbook.Names
If wbNmz.Count = 0 Then
        MsgBox "There are no names in this workbook", vbOK, g_sAPP_TITLE
        Exit Sub
End If
```

```
On Error Resume Next

Debug.Print "*****  NAMES IN THE " & ThisWorkbook.name & " WORKBOOK  *****" & _
    vbCr
Dim i As Integer
For i = 1 To wbNmz.Count
    Set wbDefNm = wbNmz(i)
    sNameNm = wbDefNm.name
    Debug.Print sNameNm & Space(30 - Len(sNameNm)) & wbDefNm.RefersTo
Next i
End Sub
'================================================================================
Public Sub PrintXlDefinedNamesToTextFile()
Dim fso As FileSystemObject
Dim ts As TextStream
Dim namesInWb As File
Dim wbNmz As Names
Dim wbDefNm As name
Dim sNameNm As String
Dim sFileNm As String
Dim iNmLen As Integer
Dim iMaxNamLen As Integer
Dim i As Integer
Dim bNameIsPrintRelated As Boolean
Dim ansOmitPrintStuff As VbMsgBoxResult

If g_cMsgBoxHandler Is Nothing Then
    Set g_cMsgBoxHandler = New MsgBoxHandler
End If
g_cMsgBoxHandler.vSetOmitPrintAreaTitlesMsg
ansOmitPrintStuff = g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbYesNo)

Set wbNmz = ThisWorkbook.Names
Set fso = New FileSystemObject
sFileNm = ThisWorkbook.FullName & "_Names.txt"
If Not fso.FileExists(sFileNm) Then
    Set ts = fso.CreateTextFile(sFileNm)
    ts.Close
End If

'Set indent...
For i = 1 To wbNmz.Count
    Set wbDefNm = wbNmz(i)
    sNameNm = wbDefNm.name
    'Set boolean...
    If ansOmitPrintStuff = vbYes Then
        bNameIsPrintRelated = _
            InStr(1, sNameNm, "!Print_Area") > 0 Or _
            InStr(1, sNameNm, "!Print_Titles") > 0
    End If
    'If we WANTED  print-related stuff boolean never reset from default(F)...
    If Not bNameIsPrintRelated Then
            iNmLen = Len(sNameNm)
        If iNmLen > iMaxNamLen Then iMaxNamLen = iNmLen
    End If
    bNameIsPrintRelated = False '...restore to default
Next i

Set namesInWb = fso.GetFile(sFileNm)
Set ts = namesInWb.OpenAsTextStream(ForWriting, TristateUseDefault)
ts.WriteLine "*****  NAMES IN THE " & ThisWorkbook.name & " WORKBOOK  *****"
If ansOmitPrintStuff = vbYes Then
    ts.Write "[OMITS PRINT-RELATED NAMES!!!]"
End If
ts.WriteBlankLines (2)
For i = 1 To wbNmz.Count
    Set wbDefNm = wbNmz(i)
    sNameNm = wbDefNm.name
    'Set boolean...
    If ansOmitPrintStuff = vbYes Then
```

```
        bNameIsPrintRelated = _
            InStr(1, sNameNm, "!Print_Area") > 0 Or _
            InStr(1, sNameNm, "!Print_Titles") > 0
    End If
    'If we WANTED  print-related stuff boolean never reset from default(F)...
    If Not bNameIsPrintRelated Then
        ts.WriteLine sNameNm & Space(iMaxNamLen + 5 - Len(sNameNm)) & wbDefNm.RefersTo
    End If
    bNameIsPrintRelated = False '...restore to default
Next i
ts.Close
Shell "Notepad.exe " & sFileNm, vbNormalFocus
End Sub
'================================================================================
Public Sub DeleteUnusedXLNames()
Dim nm As name
Dim i As Integer
For Each nm In ThisWorkbook.Names
    If StrComp(Right(nm.name, 15), "_FilterDatabase") = 0 Then
        i = i + 1
        nm.Delete
    End If
Next nm
MsgBox "Nmbr names deleted:  " & i
End Sub


'================================================================================
'================================================================================
'================================================================================
'================================================================================
```

```
'.....................................................................................
'.....................................................................................
'.......        Developer:              William H. White (consultant)
'.......        With:                   TEKsystems Inc.
'.......                                www.teksystems.com
'.......        For:                    AIG
'.......                                Financial Information Systems
'.......                                1 NY Plaza, 15th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         9/13/2013
'.......        Proj finished:          11/21/2013
'.......        Proj revised:           1/15/2014
'.....................................................................................
'.....................................................................................

'.....................................................................................
'.........................  Class-level COMMENTS  ....................................
'This is a wrapper class for dimension/vaule pairs.  I would have used _
simple Dictionary types except that I wanted to add properties to identify: _
1) The type of demension/value pairs (e.g., File, Worksheet, etc.) _
2) Source file, source worksheet, etc.
'.....................................................................................
'.....................................................................................


'*************************************************************************************
'*************************************************************************************
'*************************  CLASS-LEVEL DECLARATIONS  ************************************
'*************************************************************************************
'*************************************************************************************
Option Explicit




'*************************************************************************************
'*************************************************************************************
'*******************************  PRIVATE MEMBERS  ***********************************
'*************************************************************************************
'*************************************************************************************

'**************************************
'MODULE/CLASS-WIDE CONSTANTS
'**************************************
'Strings...
Private Const m_sMODULE_NAME As String = "ByDimenNmFrmCtrls"


'**************************************
'MODULE/CLASS-WIDE VARIABLES
'**************************************
'Collections...
Private m_coll As Collection
Private m_collKeys As Collection



'=============================================================
'======================  EVENTS  ============================
'=============================================================
'=====================================================================================
Private Sub Class_Initialize()
Set m_coll = New Collection
Set m_collKeys = New Collection
End Sub
'=====================================================================================
Private Sub Class_Terminate()
```

```
ByDimenNmFrmCtrls - 2

'Since there's really nothing useful an error-handler can do at this point...
On Error Resume Next
Set m_coll = Nothing
Set m_collKeys = Nothing
End Sub
'============================================================
'==================  (Private) FUNCTIONS  ===================
'============================================================
'================================================================================
Private Function GetNumericIndexFromStringKey(ByVal Key) As Byte
Dim iKey As Byte
Dim i As Byte
'Get the integer key by looping through the keys collection...
For i = 1 To m_collKeys.Count
    If StrComp(Trim(Key), Trim(m_collKeys(i))) = 0 Then
        iKey = i
        i = m_collKeys.Count
    End If
Next i
GetNumericIndexFromStringKey = iKey
End Function




'********************************************************************************
'********************************************************************************
'*****************************  PUBLIC MEMBERS  *********************************
'********************************************************************************
'********************************************************************************


'============================================================
'======================  PROPERTIES  ========================
'============================================================
'================================================================================
'Property Count (read-only)...
Property Get Count() As Long
    Count = m_coll.Count
End Property
'Property NewEnum (read-only)...
Public Property Get NewEnum() As IUnknown
        Set NewEnum = m_coll.[_NewEnum]
End Property


'============================================================
'==================  (Public) FUNCTIONS  ====================
'============================================================
'================================================================================
Public Function Item(index) As MSForms.control
'This is idiotic.  THE WHOLE POINT of creating collections was so that I _
could skip FOR loops and retrieve controls directly via their Dimension Names, _
which were placed in their TAGs.  However, I come to find that if I want to _
retrieve a control with "Year" as its tag I CANNOT declare, say, "key" as _
a string variable, set it = "Year", and then set Item=m_coll(key).  I can _
ONLY get the item from the collection by using a string literal - i.e., _
Item=m_coll("Year").  ARRRGGGGGHHHHHHH!!!!
'Ergo, I have to loop anyway!!!  GRRRRRRRRRRR.....
'[I CAN, however, use an INTEGER variable to retrieve an item from a _
collection.  .NET her I come!!!]
If IsNumeric(index) Then
    index = Val(index) '...just in case it is passed in as a string
    Set Item = m_coll(index)
Else
    Set Item = m_coll(GetNumericIndexFromStringKey(index))
End If
End Function
'================================================================================
Public Function Contains(ByVal DimenNm As String) As Boolean
Dim iKeys As Integer
```

```
ByDimenNmFrmCtrls - 3

Dim i As Integer
Dim bContains

iKeys = m_collKeys.Count
For i = 1 To iKeys
    If StrComp(Trim(DimenNm), m_collKeys(i), vbBinaryCompare) = 0 Then
        bContains = True
        i = iKeys '...break loop
    End If
Next i
Contains = bContains
End Function



'=============================================================
'==================  (Public) PROCEDURES  ==================
'=============================================================
'=============================================================================
Public Sub Add(VCtrl As MSForms.control)
'No need for error handling; incompatible types will not even compile...
m_coll.Add Item:=VCtrl, Key:=VCtrl.Tag
m_collKeys.Add Item:=VCtrl.Tag, Key:=VCtrl.Tag
End Sub
'=============================================================================
Public Sub Remove(KDimenNm As String, ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "Remove()"
'Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

m_coll.Remove KDimenNm
m_collKeys.Remove KDimenNm

ExitPoint:
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'=============================================================================
Public Sub RemoveAll()
Set m_coll = New Collection
Set m_collKeys = New Collection
End Sub
'=============================================================================
'=============================================================================
'=============================================================================
'=============================================================================
'=============================================================================
```

```
'.....................................................................................
'.....................................................................................
'.......        Developer:              William H. White (consultant)
'.......        With:                   TEKsystems Inc.
'.......                                www.teksystems.com
'.......        For:                    AIG
'.......                                Financial Information Systems
'.......                                1 NY Plaza, 15th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         8/21/2013
'.......        Proj finished:          11/21/2013
'.......        Proj revised:           1/15/2014
'.....................................................................................
'.....................................................................................

'.....................................................................................
'.........................  Class-level COMMENTS  ....................................
'This is a wrapper class for dimension/row or column number  pairs.  I would _
have used simple Dictionary types except that I wanted to add properties to _
identify: _
1) The type of demension/value pairs (e.g., File, Worksheet, etc.) _
2) Source file, source worksheet, etc.
'.....................................................................................
'.....................................................................................


'*************************************************************************************
'*************************************************************************************
'*************************  CLASS-LEVEL DECLARATIONS  ********************************
'*************************************************************************************
'*************************************************************************************
Option Explicit




'*************************************************************************************
'*************************************************************************************
'*******************************  PRIVATE MEMBERS  ***********************************
'*************************************************************************************
'*************************************************************************************

'**************************************
'MODULE/CLASS-WIDE CONSTANTS
'**************************************
'Strings...
Private Const m_sMODULE_NAME As String = "ByDimenRowColNos"


'**************************************
'MODULE/CLASS-WIDE VARIABLES
'**************************************
'Collections...
Private m_coll As Collection
Private m_collKeys As Collection

'Dimension Scope...
Private m_enumScope As ProjDimenScope

'Amount Type...
Private m_enumAmtType As ProjAmtType

'Strings...
Private m_sSourceWsh As String

'=========================================================
'=======================  EVENTS  ========================
```

```
ByDimenRowColNos - 2

'============================================================
'============================================================================
Private Sub Class_Initialize()
Set m_coll = New Collection
Set m_collKeys = New Collection
End Sub
'============================================================================
Private Sub Class_Terminate()
'Since there's really nothing useful an error-handler can do at this point...
On Error Resume Next
Set m_coll = Nothing
Set m_collKeys = Nothing
End Sub
'============================================================
'==================  (Private) FUNCTIONS  ===================
'============================================================
'============================================================================
Private Function GetNumericIndexFromStringKey(ByVal Key) As Byte
Dim iKey As Byte
Dim i As Byte
'Get the integer key by looping through the keys collection...
For i = 1 To m_collKeys.Count
    If StrComp(Trim(Key), Trim(m_collKeys(i))) = 0 Then
        iKey = i
        i = m_collKeys.Count
    End If
Next i
GetNumericIndexFromStringKey = iKey
End Function




'****************************************************************************
'****************************************************************************
'*****************************  PUBLIC MEMBERS  *****************************
'****************************************************************************
'****************************************************************************


'============================================================
'======================  PROPERTIES  ========================
'============================================================
'============================================================================
'DimensionScope...
Property Let DimensionScope(value As ProjDimenScope)
    m_enumScope = value
End Property
Property Get DimensionScope() As ProjDimenScope
    DimensionScope = m_enumScope
End Property
'SourceWsh...
Property Let SourceWsh(value As String)
    m_sSourceWsh = value
End Property
Property Get SourceWsh() As String
    SourceWsh = m_sSourceWsh
End Property
'AmtType...
Property Let AmountType(value As ProjAmtType)
    m_enumAmtType = value
End Property
Property Get AmountType() As ProjAmtType
    AmountType = m_enumAmtType
End Property
'Property NewEnum (read-only)...
Public Property Get NewEnum() As IUnknown
        Set NewEnum = m_coll.[_NewEnum]
End Property
```

```
ByDimenRowColNos - 3


'============================================================
'==================  (Public) FUNCTIONS  ==================
'============================================================
'============================================================================
Public Function Items() As Dictionary
Set Items = m_coll
End Function
'============================================================================
Public Function Item(index) As Long
Item = m_coll(index)
End Function
'============================================================================
Public Function Count() As Integer
Count = m_coll.Count
End Function
'============================================================================
Public Function Exists(Dimen As Dimension) As Boolean
Dim bExists As Boolean
bExists = m_coll.Exists(Dimen)
Exists = bExists
End Function
'============================================================================
Public Function Keys() As Dimension()
Keys = m_coll.Keys
End Function




'============================================================
'==================  (Public) PROCEDURES  ==================
'============================================================
'============================================================================
Public Sub Add(Dimen As Dimension, value As Long)
'No need for error handling; incompatible types will not even compile...
m_coll.Add Key:=Dimen, Item:=value
End Sub
'============================================================================
Public Sub Remove(Dimen, ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "Remove()"
'Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

m_coll.Remove Dimen
m_collKeys.Remove Dimen

ExitPoint:
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
```

```
ByDimenRowColNos - 4

'=============================================================================
Public Sub RemoveAll()
Set m_coll = New Collection
Set m_collKeys = New Collection
End Sub
'=============================================================================
Public Sub Initialize(ByRef RanOkRESULT As Boolean)
'[NOTE:  Use this procedure whenever there is start-up code which might _
possibly fail.  Put such code here, which is a procedure which must be _
called expicitly, because, as noted above, errors which might occur in _
the Class_Initialize() method CANNOT be trapped by a calling procedure.]

'Error-handling declarations...
Const sSOURCE As String = "Initialize()"
Const bPROCEDURE_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until an error is encountered...
RanOkRESULT = True

'Call a lower-level sub-routine.  Make sure the SUB has a ByRef boolean _
parameter for error-handling...
'''NonEntryPointProcedure RanOkRESULT:= bCalledProcedureRanOk
'''if not bCalledProcedureRanOk then Err.Raise g_lERR_HANDLED

ExitPoint:
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bPROCEDURE_IS_ENTRY_PT
        If bRetraceErrorMode Then
                Else
                Resume ExitPoint
        End If
End Sub
'=============================================================================
'=============================================================================
'=============================================================================
'=============================================================================
'=============================================================================
'=============================================================================
```

```
'.....................................................................................
'.....................................................................................
'.......        Developer:              William H. White (consultant)
'.......        With:                   TEKsystems Inc.
'.......                                www.teksystems.com
'.......        For:                    AIG
'.......                                Financial Information Systems
'.......                                1 NY Plaza, 15th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         11/21/2013
'.......        Proj finished:          11/21/2013
'.......        Proj revised:           1/15/2014
'.....................................................................................
'.....................................................................................

'.....................................................................................
'.........................  Class-level COMMENTS  ....................................
'Commentary...
'.....................................................................................
'.....................................................................................


'*************************************************************************************
'*************************************************************************************
'*************************  CLASS-LEVEL DECLARATIONS  ************************************
'*************************************************************************************
'*************************************************************************************
Option Explicit
Option Compare Binary '...use cAsE senSITive string comparisons



'*************************************************************************************
'*************************************************************************************
'*******************************  PRIVATE MEMBERS  ***********************************
'*************************************************************************************
'*************************************************************************************

'**************************************
'MODULE/CLASS-WIDE CONSTANTS
'**************************************
'Strings...
Private Const m_sMODULE_NAME As String = "ByNmFormControls"


'**************************************
'MODULE/CLASS-WIDE VARIABLES
'**************************************
'Collections...
Private m_coll As Collection
Private m_collKeys As Collection



'============================================================
'======================  EVENTS  ============================
'============================================================
'=============================================================================
Private Sub Class_Initialize()
Set m_coll = New Collection
Set m_collKeys = New Collection
End Sub
'=============================================================================
Private Sub Class_Terminate()
'Since there's really nothing useful an error-handler can do at this point...
On Error Resume Next
Set m_coll = Nothing
```

```
ByNmFormControls - 2

Set m_collKeys = Nothing
End Sub
'=============================================================
'==================  (Private) FUNCTIONS  ===================
'=============================================================
'=============================================================================
Private Function GetNumericIndexFromStringKey(ByVal Key) As Byte
Dim iKey As Byte
Dim i As Byte
'Get the integer key by looping through the keys collection...
For i = 1 To m_collKeys.Count
    If StrComp(Trim(Key), Trim(m_collKeys(i))) = 0 Then
        iKey = i
        i = m_collKeys.Count
    End If
Next i
GetNumericIndexFromStringKey = iKey
End Function




'*****************************************************************************
'*****************************************************************************
'*****************************  PUBLIC MEMBERS  ******************************
'*****************************************************************************
'*****************************************************************************

'=============================================================
'======================  PROPERTIES  =========================
'=============================================================
'=============================================================================
'Property NewEnum...
Public Property Get NewEnum() As IUnknown
        Set NewEnum = m_coll.[_NewEnum]
End Function

'=============================================================
'==================  (Public) FUNCTIONS  ===================
'=============================================================
'=============================================================================
Public Function Items() As Collection
Set Items = m_coll
End Function
'=============================================================================
Public Function Keys() As Collection
Set Keys = m_collKeys
End Function
'=============================================================================
Public Function Item(index) As MSForms.control
If IsNumeric(index) Then
    index = Val(index) '...just in case it is passed in as a string
    Set Item = m_coll(index)
Else
    Set Item = m_coll(GetNumericIndexFromStringKey(index))
End If
End Function
'=============================================================================
Public Function Count() As Integer
Count = m_coll.Count
End Function
'=============================================================================
Public Function ContainsItem(ItemName As String) As Boolean
Dim bContains As Boolean

If m_coll.Count > 0 Then
        Dim ctrl As MSForms.control
        Dim i As Integer
        For i = 1 To m_coll.Count
                Set ctrl = m_coll.Item(i)
                If StrComp(ctrl.name, ItemName) = 0 Then
```

```
ByNmFormControls - 3

                        bContains = True
                        Exit For
                End If
        Next i
End If
ContainsItem = bContains
End Function



'=============================================================
'==================  (Public) PROCEDURES  ==================
'=============================================================
'=============================================================================
Public Sub Add(VCtrl As MSForms.control)
'No need for error handling; incompatible types will not even compile...
m_coll.Add Item:=VCtrl, Key:=VCtrl.name
m_collKeys.Add Item:=VCtrl.name, Key:=VCtrl.name
End Sub
'=============================================================================
Public Sub Remove(index, ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "Remove()"
'Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

m_coll.Remove index
m_collKeys.Remove index

ExitPoint:
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'=============================================================================
Public Sub RemoveAll()
Set m_coll = Nothing
Set m_collKeys = New Collection
End Sub
'=============================================================================
'=============================================================================
'=============================================================================
'=============================================================================
'=============================================================================
```

```
'.....................................................................................
'.....................................................................................
'.......        Developer:              William H. White (consultant)
'.......        With:                   TEKsystems Inc.
'.......                                www.teksystems.com
'.......        For:                    AIG
'.......                                Financial Information Systems
'.......                                1 NY Plaza, 15th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         8/26/2013
'.......        Proj finished:          11/21/2013
'.......        Proj revised:           1/15/2014
'.....................................................................................
'.....................................................................................

'.....................................................................................
'.........................  Class-level COMMENTS  ....................................
'This is a wrapper class for pairs of form controls.  It is designed so _
that we can look up one control (e.g., refedYear) via another control _
(e.g., txtYear).
'.....................................................................................
'.....................................................................................


'*************************************************************************************
'*************************************************************************************
'*************************  CLASS-LEVEL DECLARATIONS  ************************************
'*************************************************************************************
'*************************************************************************************
Option Explicit




'*************************************************************************************
'*************************************************************************************
'*******************************  PRIVATE MEMBERS  ***********************************
'*************************************************************************************
'*************************************************************************************

'**************************************
'MODULE/CLASS-WIDE CONSTANTS
'**************************************
'Strings...
Private Const m_sMODULE_NAME As String = "CtrlNmByCtrlNm"


'**************************************
'MODULE/CLASS-WIDE VARIABLES
'**************************************
'Collections...
Private m_dict As Dictionary

'Strings...
Private m_sFormNm As String

'============================================================
'======================  EVENTS  ============================
'============================================================
'=============================================================================
Private Sub Class_Initialize()
Set m_dict = New Dictionary
m_dict.CompareMode = TextCompare
End Sub
'=============================================================================
Private Sub Class_Terminate()
'Since there's really nothing useful an error-handler can do at this point...
```

```
CtrlNmByCtrlNm - 2

On Error Resume Next
Set m_dict = Nothing
End Sub




'*********************************************************************************
'*********************************************************************************
'*****************************  PUBLIC MEMBERS  **********************************
'*********************************************************************************
'*********************************************************************************


'============================================================
'======================  PROPERTIES  ========================
'============================================================
'=============================================================================
'FormName...
Property Let FormName(value As String)
    m_sFormNm = value
End Property
Property Get FormName() As String
    FormName = m_sFormNm
End Property
'CompareMode (read-only)...
Property Get CompareMode() As CompareMethod
CompareMode = m_dict.CompareMode
End Property




'============================================================
'==================  (Public) FUNCTIONS  ====================
'============================================================
'=============================================================================
Public Function Items() As Dictionary
Set Items = m_dict
End Function
'=============================================================================
Public Function Item(index) As String
Item = m_dict(index)
End Function
'=============================================================================
Public Function Count() As Integer
Count = m_dict.Count
End Function
'=============================================================================
Public Function Exists(ctrlNm As String) As Boolean
Dim bExists As Boolean
bExists = m_dict.Exists(ctrlNm)
Exists = bExists
End Function
'=============================================================================
Public Function Keys() As String()
Keys = m_dict.Keys
End Function




'============================================================
'==================  (Public) PROCEDURES  ===================
'============================================================
'=============================================================================
Public Sub Add(keyCtrlName As String, valCtrlName As String)
'No need for error handling; incompatible types will not even compile...
m_dict.Add Key:=keyCtrlName, Item:=valCtrlName
End Sub
```

```
CtrlNmByCtrlNm - 3

'=============================================================================
Public Sub Remove(keyCtrlName, ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "Remove()"
'Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

m_dict.Remove keyCtrlName

ExitPoint:
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'=============================================================================
Public Sub RemoveAll()
Set m_dict = New Dictionary
End Sub
'=============================================================================
'=============================================================================
'=============================================================================
'=============================================================================
'=============================================================================
'=============================================================================
```

Dimension - 1

```
'.....................................................................................
'.....................................................................................
'.......        Developer:              William H. White (consultant)
'.......        With:                   TEKsystems Inc.
'.......                                www.teksystems.com
'.......        For:                    AIG
'.......                                Financial Information Systems
'.......                                1 NY Plaza, 15th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         8/16/2013
'.......        Proj finished:          11/21/2013
'.......        Proj revised:           1/15/2014
'.....................................................................................
'.....................................................................................


'.....................................................................................
'.........................  Class-level COMMENTS  .....................................
'I initially made Dimension a global type.  However, after several hours of _
frustration & hair-pulling I discovered that custom collection classes can _
only hold object types (i.e., an instance of a class).  So, I have made _
Dimension a class.
'.....................................................................................
'.....................................................................................


'*************************************************************************************
'*************************************************************************************
'*************************  CLASS-LEVEL DECLARATIONS  ************************************
'*************************************************************************************
'*************************************************************************************
Option Explicit
Option Compare Binary '...use cAsE senSITive string comparisons




'*************************************************************************************
'*************************************************************************************
'*******************************  PRIVATE MEMBERS  ***********************************
'*************************************************************************************
'*************************************************************************************

'***************************************
'MODULE/CLASS-WIDE CONSTANTS
'***************************************
'Strings...
Private Const m_sMODULE_NAME As String = "Dimension"



'***************************************
'MODULE/CLASS-WIDE VARIABLES
'***************************************
'Strings...
Private m_sName As String
Private m_sValListLabelXLNm As String

'ProjDimenReqs...
Private m_enumReqType As ProjDimenReq

'Bytes...
Private m_ySLFCol As Byte '...Staging Layer Feed column



'=======================================================
```

```
Dimension - 2

'======================  EVENTS  ========================
'=========================================================
'=========================================================================
Private Sub Class_Initialize()
'[VBA QUIRK:  Errors in Intialize, Activate, and Terminate event handlers _
cannot be handled by procedures which create or destroy instances of the _
class. If any such error-handling IS required, create a separate public _
procedure - e.g., the Initialize procedure, below.]
End Sub
'=========================================================================
Private Sub Class_Terminate()
'[VBA QUIRK:  Errors in Intialize, Activate, and Terminate event handlers _
cannot be handled by procedures which create or destroy instances of the _
class. If any such error-handling IS required, create a separate public _
procedure - e.g., the Initialize procedure, below.]

'Since there's really nothing useful an error-handler can do at this point...
On Error Resume Next
'clean-up code...
End Sub



'*************************************************************************
'*************************************************************************
'*****************************  PUBLIC MEMBERS  ******************************
'*************************************************************************
'*************************************************************************

'=========================================================
'======================  PROPERTIES  ======================
'=========================================================
'=========================================================================
'Property Name...
Property Let name(value As String)
    m_sName = value
End Property
Property Get name() As String
    name = m_sName
End Property
'Property RequirementType...
Property Let RequirementType(value As ProjDimenReq)
    m_enumReqType = value
End Property
Property Get RequirementType() As ProjDimenReq
    RequirementType = m_enumReqType
End Property
'Property SLFColumn...
Property Let SLFColumn(value As Byte)
    m_ySLFCol = value
End Property
Property Get SLFColumn() As Byte
    SLFColumn = m_ySLFCol
End Property
'Property ValListLabelXLName...
Property Let ValListLabelXLName(value As String)
    m_sValListLabelXLNm = value
End Property
Property Get ValListLabelXLName() As String
    ValListLabelXLName = m_sValListLabelXLNm
End Property



'=========================================================
'=================  (Public) PROCEDURES  =================
'=========================================================
'=========================================================================
Public Sub InitiateProperties(ByVal name As String, _
    ByVal ReqType As ProjDimenReq, _
```

```
Dimension - 3

    ByVal SLFColNmbr As Byte, _
    ByVal ValListLabelXLNm As String, _
    ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "InitiateProperties()"
Const bSUB_IS_ENTRY_PT As Boolean = False
'Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'Set properties...
Me.name = name
Me.RequirementType = ReqType
Me.SLFColumn = SLFColNmbr
Me.ValListLabelXLName = ValListLabelXLNm

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub


'=============================================================================
'=============================================================================
'=============================================================================
'=============================================================================
'=============================================================================
'=============================================================================
```

DimensionHandler - 1

```
'.....................................................................................
'.....................................................................................
'.......        Developer:              William H. White (consultant)
'.......        With:                   TEKsystems Inc.
'.......                                www.teksystems.com
'.......        For:                    AIG
'.......                                Financial Information Systems
'.......                                1 NY Plaza, 15th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         8/8/2013
'.......        Proj finished:          11/21/2013
'.......        Proj revised:           1/15/2014
'.....................................................................................
'.....................................................................................


'*************************************************************************************
'*************************************************************************************
'*************************  CLASS-LEVEL DECLARATIONS  ************************
'*************************************************************************************
'*************************************************************************************
Option Explicit
Option Compare Binary '...use cAsE senSITive string comparisons




'*************************************************************************************
'*************************************************************************************
'*******************************  PRIVATE MEMBERS  *******************************
'*************************************************************************************
'*************************************************************************************

'***************************************
'MODULE/CLASS-WIDE COLLECTIONS
'***************************************
Private m_AllDimens As Dimensions
Private m_MandDimens As Dimensions
Private m_FileDimens As Dimensions
Private m_WshTypes As WshTypes



'***************************************
'MODULE/CLASS-WIDE DICTIONARIES
'***************************************
Private m_dictWshDimenVals As ValueByDimen
Private m_dictRowDimenLocs As ByDimenRowColNos
Private m_dictColDimenLocs As ByDimenRowColNos


'***************************************
'MODULE/CLASS-WIDE CONSTANTS
'***************************************
'Strings...
Private Const m_sMODULE_NAME As String = "DimensionHandler"




'=============================================================
'======================  EVENTS  ============================
'=============================================================
'=============================================================================
Private Sub Class_Initialize()
'[VBA QUIRK:  Errors in Intialize, Activate, and Terminate event handlers _
```

```
DimensionHandler - 2

cannot be handled by procedures which create or destroy instances of the _
class. If any such error-handling IS required, create a separate public _
procedure - e.g., the Initialize procedure, below.]
End Sub
'==========================================================================
Private Sub Class_Terminate()
'[VBA QUIRK:  Errors in Intialize, Activate, and Terminate event handlers _
cannot be handled by procedures which create or destroy instances of the _
class. If any such error-handling IS required, create a separate public _
procedure - e.g., the Initialize procedure, below.]

'Since there's really nothing useful an error-handler can do at this point...
On Error Resume Next
'clean-up code...
End Sub




'==========================================================
'=================  (Private) PROCEDURES  ================
'==========================================================
'==========================================================================
Private Sub PopulateAllDimensionsCollection(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "PopulateAllDimensionsCollection()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim arrDimens(0 To g_yTTL_NO_DIMENS - 1) As Dimension
Dim i As Byte

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'In case the user closed the File Dimension form via Cancel button but has _
now re-opened the form...
If m_AllDimens.Count > 0 Then Set m_AllDimens = New Dimensions

'Create an array of all dimensions...
Set arrDimens(0) = g_cObjFactory.CreateDimension( _
        name:=g_sDIMEN_YR, _
        ReqEnum:=projDimenMandatory, _
        SLFCol:=g_yCOL_DIMEN_YR, _
        ValListLabelXLName:="ptrLblYear", _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set arrDimens(1) = g_cObjFactory.CreateDimension( _
        name:=g_sDIMEN_PER, _
        ReqEnum:=projDimenMandatory, _
        SLFCol:=g_yCOL_DIMEN_PER, _
        ValListLabelXLName:="ptrLblPeriod", _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set arrDimens(2) = g_cObjFactory.CreateDimension( _
        name:=g_sDIMEN_SCENAR, _
        ReqEnum:=projDimenMandatory, _
        SLFCol:=g_yCOL_DIMEN_SCENAR, _
        ValListLabelXLName:="ptrLblScenario", _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set arrDimens(3) = g_cObjFactory.CreateDimension( _
        name:=g_sDIMEN_DATA_SRC, _
        ReqEnum:=projDimenMandatory, _
        SLFCol:=g_yCOL_DIMEN_DATA_SRC, _
        ValListLabelXLName:="ptrLblDataSrc", _
        RanOkRESULT:=bCalledProcedureRanOk)
```

```
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set arrDimens(4) = g_cObjFactory.CreateDimension( _
        name:=g_sDIMEN_MV_ACCT, _
        ReqEnum:=projDimenMandatory, _
        SLFCol:=g_yCOL_DIMEN_MV_ACCT, _
        ValListLabelXLName:="ptrLblMVAcc", _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set arrDimens(5) = g_cObjFactory.CreateDimension( _
        name:=g_sDIMEN_LDGR, _
        ReqEnum:=projDimenMandatory, _
        SLFCol:=g_yCOL_DIMEN_LDGR, _
        ValListLabelXLName:="ptrLblLedger", _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set arrDimens(6) = g_cObjFactory.CreateDimension( _
        name:=g_sDIMEN_TRANS_TYP, _
        ReqEnum:=projDimenOptional, _
        SLFCol:=g_yCOL_DIMEN_TRANS_TYP, _
        ValListLabelXLName:="ptrLblTransType", _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set arrDimens(7) = g_cObjFactory.CreateDimension( _
        name:=g_sDIMEN_MGD_BUS, _
        ReqEnum:=projDimenMandatory, _
        SLFCol:=g_yCOL_DIMEN_MGD_BUS, _
        ValListLabelXLName:="ptrLblMngdBus", _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set arrDimens(8) = g_cObjFactory.CreateDimension( _
        name:=g_sDIMEN_FUNC, _
        ReqEnum:=projDimenMandatory, _
        SLFCol:=g_yCOL_DIMEN_FUNC, _
        ValListLabelXLName:="ptrFunction", _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set arrDimens(9) = g_cObjFactory.CreateDimension( _
        name:=g_sDIMEN_ORIG_GEOG, _
        ReqEnum:=projDimenMandatory, _
        SLFCol:=g_yCOL_DIMEN_ORIG_GEOG, _
        ValListLabelXLName:="ptrLblOrigGeography", _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set arrDimens(10) = g_cObjFactory.CreateDimension( _
        name:=g_sDIMEN_LGL_ENT, _
        ReqEnum:=projDimenMandatory, _
        SLFCol:=g_yCOL_DIMEN_LGL_ENT, _
        ValListLabelXLName:="ptrLblLglEntity", _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set arrDimens(11) = g_cObjFactory.CreateDimension( _
        name:=g_sDIMEN_PARTNER, _
        ReqEnum:=projDimenOptional, _
        SLFCol:=g_yCOL_DIMEN_PARTNER, _
        ValListLabelXLName:="ptrLblTrdgPartner", _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set arrDimens(12) = g_cObjFactory.CreateDimension( _
        name:=g_sDIMEN_PROD, _
        ReqEnum:=projDimenMandatory, _
        SLFCol:=g_yCOL_DIMEN_PROD, _
        ValListLabelXLName:="ptrLblProduct", _
```

```
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set arrDimens(13) = g_cObjFactory.CreateDimension( _
        name:=g_sDIMEN_CUST_TYPE, _
        ReqEnum:=projDimenOptional, _
        SLFCol:=g_yCOL_DIMEN_CUST_TYPE, _
        ValListLabelXLName:="ptrLblCustType", _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set arrDimens(14) = g_cObjFactory.CreateDimension( _
        name:=g_sDIMEN_DISTRIB, _
        ReqEnum:=projDimenOptional, _
        SLFCol:=g_yCOL_DIMEN_DISTRIB, _
        ValListLabelXLName:="ptrLblDistribCh", _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set arrDimens(15) = g_cObjFactory.CreateDimension( _
        name:=g_sDIMEN_RISK_LOC, _
        ReqEnum:=projDimenOptional, _
        SLFCol:=g_yCOL_DIMEN_RISK_LOC, _
        ValListLabelXLName:="ptrLblRiskLoc", _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set arrDimens(16) = g_cObjFactory.CreateDimension( _
        name:=g_sDIMEN_PROJ, _
        ReqEnum:=projDimenOptional, _
        SLFCol:=g_yCOL_DIMEN_PROJ, _
        ValListLabelXLName:="ptrLblProjInit", _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set arrDimens(17) = g_cObjFactory.CreateDimension( _
        name:=g_sDIMEN_ACC_YR, _
        ReqEnum:=projDimenOptional, _
        SLFCol:=g_yCOL_DIMEN_ACC_YR, _
        ValListLabelXLName:="ptrLblAccYear", _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set arrDimens(18) = g_cObjFactory.CreateDimension( _
        name:=g_sDIMEN_UND_YR, _
        ReqEnum:=projDimenOptional, _
        SLFCol:=g_yCOL_DIMEN_UND_YR, _
        ValListLabelXLName:="ptrLblUndYear", _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set arrDimens(19) = g_cObjFactory.CreateDimension( _
        name:=g_sDIMEN_RSVD1, _
        ReqEnum:=projDimenReserved, _
        SLFCol:=g_yCOL_DIMEN_RSVD1, _
        ValListLabelXLName:="ptrLblRsvd1", _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set arrDimens(20) = g_cObjFactory.CreateDimension( _
        name:=g_sDIMEN_RSVD2, _
        ReqEnum:=projDimenReserved, _
        SLFCol:=g_yCOL_DIMEN_RSVD2, _
        ValListLabelXLName:="ptrLblRsvd2", _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set arrDimens(21) = g_cObjFactory.CreateDimension( _
        name:=g_sDIMEN_CURR_TRANS, _
        ReqEnum:=projDimenOptional, _
        SLFCol:=g_yCOL_DIMEN_TRANS_CURR, _
```

```
        ValListLabelXLName:="ptrLblCurrTrans", _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set arrDimens(22) = g_cObjFactory.CreateDimension( _
        name:=g_sDIMEN_CURR_FUNC, _
        ReqEnum:=projDimenOptional, _
        SLFCol:=g_yCOL_DIMEN_FUNC_CURR, _
        ValListLabelXLName:="ptrLblCurrFunc", _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set arrDimens(23) = g_cObjFactory.CreateDimension( _
        name:=g_sDIMEN_CURR_LOC, _
        ReqEnum:=projDimenOptional, _
        SLFCol:=g_yCOL_DIMEN_LOC_CURR, _
        ValListLabelXLName:="ptrLblCurrLocal", _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set arrDimens(24) = g_cObjFactory.CreateDimension( _
        name:=g_sDIMEN_OPER_LNS, _
        ReqEnum:=projDimenMandatory, _
        SLFCol:=g_yCOL_DIMEN_OPER_LNS, _
        ValListLabelXLName:="ptrLblOpLines", _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

'Add dimensions to collection class...
For i = 1 To g_yTTL_NO_DIMENS
    m_AllDimens.Add arrDimens(i - 1)
Next i

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub




'*****************************************************************************
'*****************************************************************************
'*****************************  PUBLIC MEMBERS  ******************************
'*****************************************************************************
'*****************************************************************************

'============================================================
'======================  PROPERTIES  ========================
'============================================================
'=============================================================================
'Property WorksheetTypes...
Public Property Get WorksheetTypes() As WshTypes
```

```
DimensionHandler - 6

    Set WorksheetTypes = m_WshTypes
End Property
'We have to do a Property Set so that we can clone the collection from _
the collection in frmTypeWshAssoc...
Public Property Let WorksheetTypes(NewColl As WshTypes)
    Set m_WshTypes = NewColl
End Property
'Property AllDimensions (read-only)...
Public Property Get AllDimensions() As Dimensions
    Set AllDimensions = m_AllDimens
End Property
'Property MandDimensions (read-only)...
Public Property Get MandDimensions() As Dimensions
    Set MandDimensions = m_MandDimens
End Property
'property FileDimensions (read-only)...
Public Property Get FileDimensions() As Dimensions
    Set FileDimensions = m_FileDimens
End Property


'============================================================
'=================  (Public) FUNCTIONS  ====================
'============================================================
'================================================================================
Public Function AnyWshTypesWithUndefinedMandDimens( _
        ByRef RanOkRESULT As Boolean) As Boolean
'Error-handling declarations...
Const sSOURCE As String = "AnyWshTypesWithUndefinedMandDimens()"
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wsValLists As Worksheet
Dim rngWshTypeInfoTbl As Range
Dim i As Byte
Dim bWshTypeWithUndefMandDimens As Boolean

'CODE HERE...
On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

Set wsValLists = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
Set rngWshTypeInfoTbl = wsValLists.Range("valtbldodynWshTypeInfo")
For i = 1 To rngWshTypeInfoTbl.Rows.Count
    If Not rngWshTypeInfoTbl.Cells(i, g_yWSH_TYPE_TBL_DEFINED_COL) Then
        bWshTypeWithUndefMandDimens = True
        i = rngWshTypeInfoTbl.Rows.Count '...break loop
    End If
Next i

AnyWshTypesWithUndefinedMandDimens = bWshTypeWithUndefMandDimens

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Function

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bFUNCTION_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
```

```
DimensionHandler - 7

                Resume ExitPoint
        End If
End Function
'================================================================================
Public Function AnyWshTypeDimensDefined(ByRef RanOkRESULT As Boolean) As Boolean
'Error-handling declarations...
Const sSOURCE As String = "AnyWshTypeDimensDefined()"
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Const yNON_WSHTYPE_DIMEN_LOC_COLS As Byte = 5
Dim wsDimenLocs As Worksheet
Dim tblDimenLocs As ListObject
Dim rngAllWshTypeCols As Range
Dim yTblCols As Byte
Dim bWshTypeDimensDefined As Boolean

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Set wsDimenLocs = g_wbSrcData.Worksheets(g_sWSH_DIMEN_LOCS)
Set tblDimenLocs = wsDimenLocs.ListObjects("tblDimenLocs")
yTblCols = tblDimenLocs.DataBodyRange.Columns.Count

Set rngAllWshTypeCols = tblDimenLocs.DataBodyRange
Set rngAllWshTypeCols = _
        rngAllWshTypeCols.Resize(, yTblCols - yNON_WSHTYPE_DIMEN_LOC_COLS)
Set rngAllWshTypeCols = rngAllWshTypeCols.Offset(0, yNON_WSHTYPE_DIMEN_LOC_COLS)

If WorksheetFunction.CountA(rngAllWshTypeCols) > 0 Then _
    bWshTypeDimensDefined = True
AnyWshTypeDimensDefined = bWshTypeDimensDefined

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Function

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
            ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bFUNCTION_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Function
'================================================================================
Public Function AnyFileDimensDefined(ByRef RanOkRESULT As Boolean) As Boolean
'Error-handling declarations...
Const sSOURCE As String = "AnyFileDimensDefined()"
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Const yNON_WSHTYPE_DIMEN_LOC_COLS As Byte = 5
Dim wsDimenLocs As Worksheet
Dim tblDimenLocs As ListObject
Dim rngFileDimensCols As Range
Dim bFileDimensDefined As Boolean

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
```

```
DimensionHandler - 8

RanOkRESULT = True

'CODE HERE...
Set wsDimenLocs = g_wbSrcData.Worksheets(g_sWSH_DIMEN_LOCS)
Set tblDimenLocs = wsDimenLocs.ListObjects("tblDimenLocs")
Set rngFileDimensCols = _
        Application.Union(tblDimenLocs.DataBodyRange.Columns(g_yDIM_TBL_FD_HC_COL), _
        tblDimenLocs.DataBodyRange.Columns(g_yDIM_TBL_FD_RR_COL))

If WorksheetFunction.CountA(rngFileDimensCols) > 0 Then _
    bFileDimensDefined = True
AnyFileDimensDefined = bFileDimensDefined

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Function

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
            ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bFUNCTION_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Function




'============================================================
'==================  (Public) PROCEDURES  ==================
'============================================================
'================================================================================
Public Sub Initialize(ByRef RanOkRESULT As Boolean)
'[NOTE:  Use this procedure whenever there is start-up code which might _
possibly fail.  Put such code here, which is a procedure which must be _
called expicitly, because, as noted above, errors which might occur in _
the Class_Initialize() method CANNOT be trapped by a calling procedure.]

'Error-handling declarations...
Const sSOURCE As String = "Initialize()"
Const bPROCEDURE_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim objDimen As Dimension

On Error GoTo ErrorHandler
'Assume success until an error is encountered...
RanOkRESULT = True

Set m_AllDimens = New Dimensions
Set m_MandDimens = New Dimensions
Set m_FileDimens = New Dimensions

PopulateAllDimensionsCollection RanOkRESULT:=bCalledProcedureRanOk
    If Not RanOkRESULT Then Err.Raise g_lERR_HANDLED
'Populate mandatory dimensions...
For Each objDimen In m_AllDimens
    If objDimen.RequirementType = projDimenMandatory Then _
        m_MandDimens.Add objDimen
Next objDimen
```

```
DimensionHandler - 9


If m_WshTypes Is Nothing Then
    Set m_WshTypes = _
            g_cObjFactory.CreateWshTypesColl(RanOkRESULT:=bCalledProcedureRanOk)
        If Not RanOkRESULT Then Err.Raise g_lERR_HANDLED
End If

Set m_dictWshDimenVals = New ValueByDimen
m_dictWshDimenVals.DimensionScope = projScopeWsh
Set m_dictRowDimenLocs = New ByDimenRowColNos
m_dictRowDimenLocs.DimensionScope = projScopeRow
Set m_dictColDimenLocs = New ByDimenRowColNos
m_dictColDimenLocs.DimensionScope = projScopeCol

ExitPoint:
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bPROCEDURE_IS_ENTRY_PT
        If bRetraceErrorMode Then
                Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Public Sub PopulateDimenTableWithValue( _
    ByVal DimenScope As ProjDimenScope, _
    ByVal TextInput As Boolean, _
    ByVal DimenName As String, _
    ByVal ValueToStore As String, _
    ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "PopulateDimenTableWithValue()"
Const bSUB_IS_ENTRY_PT As Boolean = False
'Operating code declarations...
Dim wsDimenLocs As Worksheet
Dim tblDimenLocs As ListObject
Dim rngDataCell As Range
Dim sTblColAddr As String
Dim sLblXLNm As String
Dim bPseudoDimen As Boolean

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
DimenName = Trim(DimenName)
ValueToStore = Trim(ValueToStore)
If StrComp(DimenName, g_sDIMEN_SCALE) = 0 Then bPseudoDimen = True

Set wsDimenLocs = g_wbSrcData.Worksheets(g_sWSH_DIMEN_LOCS)
Set tblDimenLocs = wsDimenLocs.ListObjects("tblDimenLocs")

'NOTE:  chr(91) = left bracket, i.e. "["
'       chr(93) = right bracket, i.e., "]"
'       chr(44) = comma, i.e., ","

'While XL tables are great for dealing with columns of data, they are _
not so good with row data.  Ergo, I had to get a little creative here...
If bPseudoDimen Then
    sLblXLNm = "ptrLblAmtScale"
Else
    sLblXLNm = m_AllDimens(DimenName).ValListLabelXLName
End If
```

```
DimensionHandler - 10

Select Case DimenScope
    Case projScopeFile
        If TextInput Then
            sTblColAddr = _
                    tblDimenLocs.name & Chr(91) & _
                    g_sDIM_LOC_TBL_HDR_FILE_HC & Chr(93)
        Else
            sTblColAddr = _
                    tblDimenLocs.name & Chr(91) & _
                    g_sDIM_LOC_TBL_HDR_FILE_RR & Chr(93)
        End If
    Case Else
        sTblColAddr = tblDimenLocs.name & Chr(91) & DimenName & Chr(93)
End Select

Set rngDataCell = Application.Intersect( _
    wsDimenLocs.Range(sTblColAddr), _
    wsDimenLocs.Range(sLblXLNm).EntireRow)

'XL seems to lop off the leading apostrophe, so we put it back on...
If Not bPseudoDimen Then ValueToStore = Chr(39) & ValueToStore

rngDataCell = ValueToStore

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Public Sub PopulateWbkDimenFormFmSrcWbkDimenTbl(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "PopulateWbkDimenFormFmSrcWbkDimenTbl()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim objDimen As Dimension
Dim wsDimenLocs As Worksheet
Dim tblDimenLocs As ListObject
Dim rngDimenNmz As Range
Dim rngDimenMandatory As Range
Dim rngDimenHardCoding As Range
Dim rngDimenRngRefs As Range
Dim c As Range
Dim sDimenNm As String
Dim yRowInRng As Byte

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'NOTE:  chr(91) = left bracket, i.e. "["
'       chr(93) = right bracket, i.e., "]"
'       chr(44) = comma, i.e., ","
```

```
DimensionHandler - 11

'CODE HERE...
Set wsDimenLocs = _
    g_wbSrcData.Worksheets(g_sWSH_DIMEN_LOCS)
Set tblDimenLocs = wsDimenLocs.ListObjects("tblDimenLocs")
Set rngDimenNmz = wsDimenLocs.Range(tblDimenLocs.name & Chr(91) & _
    g_sDIM_LOC_TBL_HDR_DIM_NM & Chr(93))
Set rngDimenMandatory = wsDimenLocs.Range(tblDimenLocs.name & Chr(91) & _
    g_sDIM_LOC_TBL_HDR_MAND & Chr(93))
Set rngDimenHardCoding = wsDimenLocs.Range(tblDimenLocs.name & Chr(91) & _
    g_sDIM_LOC_TBL_HDR_FILE_HC & Chr(93))
Set rngDimenRngRefs = wsDimenLocs.Range(tblDimenLocs.name & Chr(91) & _
    g_sDIM_LOC_TBL_HDR_FILE_RR & Chr(93))

'Populate hard-coded file dimensions...
If g_bStepThruMode Then Stop
yRowInRng = 1
For Each c In rngDimenHardCoding
    If StrComp(Trim(c), vbNullString) <> 0 Then
        If Not g_bMandDimensOnly Or _
                (g_bMandDimensOnly And _
                wsDimenLocs.Cells(c.Row, g_yDIM_TBL_MAND_COL)) Then
            sDimenNm = Trim(rngDimenNmz.Cells(yRowInRng, 1))
            g_cFormsHandler.PopulateDimenIDFormWithValueFromSrcWbkDimenTbl _
                    DimScope:=projScopeFile, _
                    ControlType:=projCtrlTypTxtBox, _
                    DimenName:=sDimenNm, _
                    DimenValue:=c, _
                    RanOkRESULT:=bCalledProcedureRanOk
                If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
            'Now log which dimensions are set on open, trapping first for _
            scale and then for amount type being set...
            If StrComp(sDimenNm, g_sDIMEN_SCALE, vbTextCompare) = 0 Then
                'We ignore if Scale is set to "Varies By Worksheet"...
                If StrComp(Trim(c), g_sSCALE_SEE_WSH, vbTextCompare) <> 0 Then
                    g_cFormsHandler.PopulateDimensOnOpenCollectionInWbkDimenForm _
                            RanOkRESULT:=bCalledProcedureRanOk, _
                            ScaleSet:=True
                        If Not bCalledProcedureRanOk Then _
                                Err.Raise g_lERR_HANDLED
                End If
            ElseIf StrComp(sDimenNm, g_sDIMEN_AMT_TYPE, vbTextCompare) = 0 Then
                g_cFormsHandler.PopulateDimensOnOpenCollectionInWbkDimenForm _
                        RanOkRESULT:=bCalledProcedureRanOk, _
                        CurrSet:=True
                    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
            Else
                Set objDimen = Me.AllDimensions.Item(sDimenNm)
                g_cFormsHandler.PopulateDimensOnOpenCollectionInWbkDimenForm _
                        RanOkRESULT:=bCalledProcedureRanOk, _
                        Dimen:=objDimen
                    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
            End If
        End If
    End If
    yRowInRng = yRowInRng + 1
Next c

'Populate range reference file dimensions...
yRowInRng = 1
If g_bStepThruMode Then Stop
For Each c In rngDimenRngRefs
    If StrComp(Trim(c), vbNullString) <> 0 Then
        If Not g_bMandDimensOnly Or _
                (g_bMandDimensOnly And _
                wsDimenLocs.Cells(c.Row, g_yDIM_TBL_MAND_COL)) Then
            sDimenNm = rngDimenNmz.Cells(yRowInRng, 1)
            g_cFormsHandler.PopulateDimenIDFormWithValueFromSrcWbkDimenTbl _
                    DimScope:=projScopeFile, _
                    ControlType:=projCtrlTypeRefEdit, _
                    DimenName:=sDimenNm, _
                    DimenValue:=c, _
```

```
DimensionHandler - 12

                    RanOkRESULT:=bCalledProcedureRanOk
                If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
            'No need for Scale and Amt Type handling here, as they are treated _
            as hard-coded dimensions...
            If g_bStepThruMode Then Stop
            Set objDimen = Me.AllDimensions.Item(sDimenNm)
            g_cFormsHandler.PopulateDimensOnOpenCollectionInWbkDimenForm _
                    RanOkRESULT:=bCalledProcedureRanOk, _
                    Dimen:=objDimen
                If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        End If
    End If
    yRowInRng = yRowInRng + 1
Next c

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Public Sub PopulateWshTypesDimenFormFmSrcWbkDimenTbl( _
    ByVal WshTypeNm As String, _
    ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "PopulateWshTypesDimenFormFmSrcWbkDimenTbl()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim enumDimenScope As ProjDimenScope
Dim wsDimenLocs As Worksheet
Dim tblDimenLocs As ListObject
Dim wsFirstWshInType As Worksheet
Dim rngDimenNmz As Range
Dim rngDimenMandatory As Range
Dim rngFileDimensHardCoded As Range
Dim rngFileDimensRngRefs As Range
Dim rngWsTypeDimenRngRefs As Range
Dim rngRangeRef As Range
Dim c As Range
Dim sDimenNm As String
Dim lNmzToMandColOffset As Long
Dim yRowInRng As Byte

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'NOTE:  chr(91) = left bracket, i.e. "["
'       chr(93) = right bracket, i.e., "]"
'       chr(44) = comma, i.e., ","

If g_bStepThruMode Then Stop
'CODE HERE...
Set wsDimenLocs = g_wbSrcData.Worksheets(g_sWSH_DIMEN_LOCS)
```

```
DimensionHandler - 13

Set tblDimenLocs = wsDimenLocs.ListObjects("tblDimenLocs")
Set rngDimenNmz = wsDimenLocs.Range(tblDimenLocs.name & Chr(91) & _
    g_sDIM_LOC_TBL_HDR_DIM_NM & Chr(93))
Set rngDimenMandatory = wsDimenLocs.Range(tblDimenLocs.name & Chr(91) & _
    g_sDIM_LOC_TBL_HDR_MAND & Chr(93))
Set rngFileDimensHardCoded = wsDimenLocs.Range(tblDimenLocs.name & Chr(91) & _
    g_sDIM_LOC_TBL_HDR_FILE_HC & Chr(93))
Set rngFileDimensRngRefs = wsDimenLocs.Range(tblDimenLocs.name & Chr(91) & _
    g_sDIM_LOC_TBL_HDR_FILE_RR & Chr(93))
Set rngWsTypeDimenRngRefs = wsDimenLocs.Range(tblDimenLocs.name & Chr(91) & _
    WshTypeNm & Chr(93))
lNmzToMandColOffset = rngDimenMandatory.Column - rngDimenNmz.Column
'We want to define/test range references within a worksheet that's in the _
WorksheetType.  Any worksheet will do, so we pick the 1st one...
Set wsFirstWshInType = Me.WorksheetTypes(WshTypeNm).Item(1)
wsFirstWshInType.Activate

If g_bStepThruMode Then Stop
'Populate the form with file hard-coded dimensions...
yRowInRng = 1
For Each c In rngFileDimensHardCoded
    If StrComp(Trim(c), vbNullString) <> 0 Then
        If Not g_bMandDimensOnly Or (g_bMandDimensOnly And _
                c.Offset(0, -g_yDT_COL_OFF_MAND_2_FD_HC)) Then
            sDimenNm = Trim(rngDimenNmz.Cells(yRowInRng, g_yWSH_TYPE_TBL_NM_COL))
            'Handle our pseudo-dimensions...
            If StrComp(sDimenNm, g_sDIMEN_SCALE, vbTextCompare) = 0 Then
                If StrComp(c, vbNullString) <> 0 Then
                    Dim frmWTD As frmWshTypeDimens
                    Set frmWTD = _
                            g_cFormsHandler.WshTypeDimensFormsColl.Item(WshTypeNm)
                    frmWTD.lblFileDimScale = c
                    frmWTD.cmbScale.enabled = False
                    frmWTD.BackColor = g_lFIELD_COLOR_DISABLED
                End If
            Else
                g_cFormsHandler.PopulateWTDFormWithFileDimenAndDisableCtrls _
                        WshTypeName:=WshTypeNm, _
                        DimenNm:=sDimenNm, _
                        DimenValue:=c, _
                        RanOkRESULT:=bCalledProcedureRanOk
                    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
            End If
        End If
    End If
    yRowInRng = yRowInRng + 1
Next c

If g_bStepThruMode Then Stop
'Do the same for file ref edit dimensions...
yRowInRng = 1
For Each c In rngFileDimensRngRefs
    If StrComp(Trim(c), vbNullString) <> 0 Then
        If Not g_bMandDimensOnly Or (g_bMandDimensOnly And _
                c.Offset(0, -g_yDT_COL_OFF_MAND_2_FD_RR)) Then
            sDimenNm = rngDimenNmz.Cells(yRowInRng, g_yWSH_TYPE_TBL_NM_COL)
            g_cFormsHandler.PopulateWTDFormWithFileDimenAndDisableCtrls _
                    WshTypeName:=WshTypeNm, _
                    DimenNm:=sDimenNm, _
                    DimenValue:=c, _
                    RanOkRESULT:=bCalledProcedureRanOk
                If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        End If
    End If
    yRowInRng = yRowInRng + 1
Next c

If g_bStepThruMode Then Stop
'Populate the form based on entries in the File Dimension table...
yRowInRng = 1
For Each c In rngWsTypeDimenRngRefs
```

```
DimensionHandler - 14

    If StrComp(Trim(c), vbNullString) <> 0 Then
        If Not g_bMandDimensOnly Or _
                (g_bMandDimensOnly And rngDimenMandatory.Cells(yRowInRng, 1)) Then
            sDimenNm = Trim(rngDimenNmz.Cells(yRowInRng, g_yWSH_TYPE_TBL_NM_COL))
            'Trap for whether we have a real dimension or a pseudo-dimension...
            If StrComp(sDimenNm, g_sDIMEN_SCALE, vbTextCompare) <> 0 Then
                Set rngRangeRef = wsFirstWshInType.Range(c)
                'Go to next c if the range ref is not of the type we want...
                enumDimenScope = projScopeWsh '...default (i.e., single-cell range)
                If rngRangeRef.Columns.Count > 1 Then enumDimenScope = projScopeCol
                If rngRangeRef.Rows.Count > 1 Then enumDimenScope = projScopeRow
            Else
                'We are working with a pseudo-dimension, which we treat as _
                a worksheet-level dimension...
                enumDimenScope = projScopeWsh
            End If
            g_cFormsHandler.PopulateDimenIDFormWithValueFromSrcWbkDimenTbl _
                    DimScope:=enumDimenScope, _
                    ControlType:=projCtrlTypeRefEdit, _
                    DimenName:=sDimenNm, _
                    DimenValue:=Trim(c), _
                    RanOkRESULT:=bCalledProcedureRanOk, _
                    WshTypeName:=WshTypeNm
                If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        Else '...mandatory AND optional dimensions
                '...to be coded
        End If
    End If
yRowInRng = yRowInRng + 1
Next c

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'=============================================================================
Public Sub CreateWshTypesFromFile(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "CreateWshTypesFromFile()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim objWshType As WshType
Dim wsValLists As Worksheet
Dim rngdoWshTypeTbl As Range
Dim rngdoWshTypeWshNmMap As Range
Dim sWshType As String
Dim i As Byte
Dim j As Byte

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True
```

```
DimensionHandler - 15

'CODE HERE...
If g_bStepThruMode Then Stop
Set wsValLists = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
Set rngdoWshTypeTbl = wsValLists.Range("valtbldodynWshTypeInfo")
Set rngdoWshTypeWshNmMap = wsValLists.Range("valtbldodynWshNmTypeMap")

'Nothing else to do if no wshtypes exist...
If StrComp(Trim(rngdoWshTypeTbl.Cells(1, g_yWSH_TYPE_TBL_NM_COL)), _
        vbNullString) = 0 Then GoTo ExitPoint

'Blow out WshTypes if they already exist...
If m_WshTypes.Count > 0 Then m_WshTypes.RemoveAll

'Add wsh type to WshTypes collection...
For i = 1 To rngdoWshTypeTbl.Rows.Count
    sWshType = Trim(rngdoWshTypeTbl.Cells(i, g_yWSH_TYPE_TBL_NM_COL))
    Set objWshType = g_cObjFactory.CreateWshType( _
            WshTypeName:=sWshType, _
            RanOkRESULT:=bCalledProcedureRanOk)
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        'Now see if there are any mappings to post...
        For j = 1 To rngdoWshTypeWshNmMap.Rows.Count
            If StrComp(sWshType, _
                    Trim(rngdoWshTypeWshNmMap.Cells(j, _
                            g_yWSH_ASSOC_TBL_TYPE_COL))) = 0 Then
                objWshType.Add g_wbSrcData.Worksheets( _
                    Trim(rngdoWshTypeWshNmMap.Cells(j, _
                            g_yWSH_ASSOC_TBL_WSH_NM_COL)))
            End If
        Next j
    m_WshTypes.Add objWshType
Next i

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'============================================================================
Public Sub SetWshTypeDimensDefinedFlagInWshTypeTable( _
    ByVal WshTypeName As String, _
    ByVal AllMandDimsDefined As Boolean, _
    ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "SetWshTypeDimensDefinedFlagInWshTypeTable()"
Const bSUB_IS_ENTRY_PT As Boolean = False
'Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wsValLists As Worksheet
Dim rngWshTypeInfoTbl As Range
Dim i As Byte

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True
```

```
DimensionHandler - 16


'CODE HERE...
WshTypeName = Trim(WshTypeName)
Set wsValLists = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
Set rngWshTypeInfoTbl = wsValLists.Range("valtbldodynWshTypeInfo")
For i = 1 To rngWshTypeInfoTbl.Rows.Count
    If StrComp(WshTypeName, _
            Trim(rngWshTypeInfoTbl.Cells(i, g_yWSH_TYPE_TBL_NM_COL))) = 0 Then
        rngWshTypeInfoTbl.Cells(i, g_yWSH_TYPE_TBL_DEFINED_COL) = _
                AllMandDimsDefined
        i = rngWshTypeInfoTbl.Rows.Count '...break loop
    End If
Next i

'Resort, by Defined, then by name...
rngWshTypeInfoTbl.Sort _
        key1:=rngWshTypeInfoTbl.Cells(1, g_yWSH_TYPE_TBL_DEFINED_COL), _
        order1:=xlAscending, _
        key2:=rngWshTypeInfoTbl.Cells(1, g_yWSH_TYPE_TBL_NM_COL), _
        order2:=xlAscending, _
        Header:=xlNo


ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'============================================================================
Public Sub PopulateWshTypesCollFromSrcWbkTbl(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "PopulateWshTypesCollFromSrcWbkTbl()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim objWshType As WshType
Dim wsValLists As Worksheet
Dim wsSrcData As Worksheet
Dim rngWshTypeInfoTbl As Range
Dim rngWshNmWshTypMapTbl As Range
Dim sWshTypeNm As String
Dim sWshNm As String
Dim i As Byte

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Set wsValLists = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
Set rngWshTypeInfoTbl = wsValLists.Range("valtbldodynWshTypeInfo")
Set rngWshNmWshTypMapTbl = wsValLists.Range("valtbldodynWshNmTypeMap")

'Just to be safe...
Me.WorksheetTypes.RemoveAll
```

```
'First populate the WshTypes collection...
For i = 1 To rngWshTypeInfoTbl.Rows.Count
    sWshTypeNm = rngWshTypeInfoTbl.Cells(i, g_yWSH_TYPE_TBL_NM_COL)
    Set objWshType = g_cObjFactory.CreateWshType( _
            WshTypeName:=sWshTypeNm, _
            RanOkRESULT:=bCalledProcedureRanOk)
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    Me.WorksheetTypes.Add objWshType
Next i

'Now populate each worsheet type with its associated worksheet(s)...
For i = 1 To rngWshNmWshTypMapTbl.Rows.Count
    sWshTypeNm = Trim(rngWshNmWshTypMapTbl.Cells(i, g_yWSH_ASSOC_TBL_TYPE_COL))
    sWshNm = Trim(rngWshNmWshTypMapTbl.Cells(i, g_yWSH_ASSOC_TBL_WSH_NM_COL))
    'If opening a virgin workbook we will have the WshType "All" but we won't _
    have any worksheets assigned to that type.  Ergo, we need to trap for that _
    within our FOR loop...
    If StrComp(sWshNm, vbNullString) <> 0 Then
        Set wsSrcData = g_wbSrcData.Worksheets(sWshNm)
        Set objWshType = Me.WorksheetTypes.Item(sWshTypeNm)
        objWshType.Add wsSrcData
    End If
Next i

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Public Sub RemoveDimenDefinitionFmAllWshTypes( _
        ByVal DimenNm As String, _
        ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "RemoveDimenDefinitionFmAllWshTypes()"
Const bSUB_IS_ENTRY_PT As Boolean = False
'''Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wsValLists As Worksheet
Dim wsDimenLocs As Worksheet
Dim rngDimenNmz As Range
Dim rngWshTypes As Range
Dim rngWshTypeDefins As Range
Dim rngCellToClear As Range
Dim rngDimenRow As Range
Dim vaWshTypes() As Variant
Dim yDimens As Byte
Dim yWshTypes As Byte
Dim i As Byte

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True
```

```
DimensionHandler - 18

'CODE HERE...
DimenNm = Trim(DimenNm)

Set wsValLists = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
Set wsDimenLocs = g_wbSrcData.Worksheets(g_sWSH_DIMEN_LOCS)
Set rngWshTypes = wsValLists.Range("vallstdodynWshTypes")
yWshTypes = rngWshTypes.Rows.Count
Set rngDimenNmz = _
        wsDimenLocs.Range("tblDimenLocs" & Chr(91) & _
        g_sDIM_LOC_TBL_HDR_DIM_NM & Chr(93))
yDimens = rngDimenNmz.Rows.Count

'Find the row we need...
For i = 1 To yDimens
    If StrComp(DimenNm, Trim(rngDimenNmz.Cells(i, 1)), vbTextCompare) = 0 Then
        Set rngDimenRow = rngDimenNmz.Cells(i, 1).EntireRow
        i = yDimens '...break loop
    End If
Next i

'Get a list of our WshTypes...
If rngWshTypes.Rows.Count = 1 Then
    ReDim vaWshTypes(1 To 1, 1 To 1)
    vaWshTypes(1, 1) = rngWshTypes.value
Else
    vaWshTypes = rngWshTypes.value
End If

For i = 1 To yWshTypes
    Set rngWshTypeDefins = _
            wsDimenLocs.Range("tblDimenLocs" & Chr(91) & _
            vaWshTypes(i, 1) & Chr(93))
    Set rngCellToClear = Application.Intersect(rngWshTypeDefins, rngDimenRow)
    rngCellToClear.ClearContents
Next i

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
            ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'============================================================================
Public Sub ClearFileDimensCollection(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "ClearFileDimensCollection()"
Const bSUB_IS_ENTRY_PT As Boolean = False
'''Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
m_FileDimens.RemoveAll
```

```
DimensionHandler - 19


ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
            ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'================================================================================
Public Sub AsNeededUpdateMandDimenDefinedFlagForWshTypes( _
        ScaleSetAtFileLevel As Boolean, _
        ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "AsNeededUpdateMandDimenDefinedFlagForWshTypes()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim dimensWshTypeUndefMand As Dimensions
Dim wsDimenLocs As Worksheet
Dim wsValLists As Worksheet
Dim tblDimenLocs As ListObject
Dim rngWshTypeStatusTbl As Range
Dim rngWshTypeDimenData As Range
Dim sWshType As String
Dim sDimen As String
Dim iWshTypeToDimenNmOffset As Integer
Dim iWshTypeToMandOffset As Integer
Dim i As Byte
Dim j As Byte
Dim bAllMandDimensDefined As Boolean

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Set wsDimenLocs = g_wbSrcData.Worksheets(g_sWSH_DIMEN_LOCS)
Set wsValLists = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
Set tblDimenLocs = wsDimenLocs.ListObjects("tblDimenLocs")
Set rngWshTypeStatusTbl = wsValLists.Range("valtbldodynWshTypeInfo")

For i = 1 To rngWshTypeStatusTbl.Rows.Count
    sWshType = Trim(rngWshTypeStatusTbl.Cells(i, g_yWSH_TYPE_TBL_NM_COL))
    bAllMandDimensDefined = True
    'We are only concerned with WshTypes flagged as not having all mandatory _
    dimensions defined...
    If Not rngWshTypeStatusTbl.Cells(i, g_yWSH_TYPE_TBL_DEFINED_COL) Then
        Set rngWshTypeDimenData = wsDimenLocs.Range(tblDimenLocs.name & _
                Chr(91) & sWshType & Chr(93))
        iWshTypeToDimenNmOffset = g_yDIM_TBL_NM_COL - rngWshTypeDimenData.Column
        iWshTypeToMandOffset = g_yDIM_TBL_MAND_COL - rngWshTypeDimenData.Column
        For j = 1 To rngWshTypeDimenData.Rows.Count
            'See if the dimension is NOT defined at the WshType level...
            If StrComp(Trim(rngWshTypeDimenData.Cells(j, 1)), _
                    vbNullString) = 0 Then
                'Now see if this is a mandatory dimension...
                If rngWshTypeDimenData.Cells(j, 1).Offset( _
```

```
                    0, iWshTypeToMandOffset) Then
                'See if the dimension is defined at the file level..
                sDimen = Trim(rngWshTypeDimenData.Cells(j, 1).Offset(0, _
                        iWshTypeToDimenNmOffset))
                'Trap for Scale...
                If StrComp(sDimen, g_sDIMEN_SCALE, vbTextCompare) = 0 Then
                    If Not ScaleSetAtFileLevel Then bAllMandDimensDefined = False
                Else '...a "regular" dimension
                    If Not Me.FileDimensions.ContainsItem( _
                            sDimen, vbTextCompare) Then
                        bAllMandDimensDefined = False
                        j = rngWshTypeDimenData.Rows.Count '...break loop
                    End If
                End If
            End If
        End If
    Next j
End If
rngWshTypeStatusTbl.Cells(i, g_yWSH_TYPE_TBL_DEFINED_COL) = _
        bAllMandDimensDefined
Next i


ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
            ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub


'================================================================================
'================================================================================
'================================================================================
'================================================================================
```

```
Dimensions - 1

'.....................................................................................
'.....................................................................................
'.......        Developer:              William H. White (consultant)
'.......        With:                   TEKsystems Inc.
'.......                                www.teksystems.com
'.......        For:                    AIG
'.......                                Financial Information Systems
'.......                                1 NY Plaza, 15th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         8/8/2013
'.......        Proj finished:          11/21/2013
'.......        Proj revised:           1/15/2014
'.....................................................................................
'.....................................................................................

'.....................................................................................
'.........................  Class-level COMMENTS  .....................................
'Commentary...
'.....................................................................................
'.....................................................................................


'*************************************************************************************
'*************************************************************************************
'*************************  CLASS-LEVEL DECLARATIONS  ************************************
'*************************************************************************************
'*************************************************************************************
Option Explicit
Option Compare Binary '...use cAsE senSITive string comparisons
Implements ICloneable



'*************************************************************************************
'*************************************************************************************
'*******************************  PRIVATE MEMBERS  ***********************************
'*************************************************************************************
'*************************************************************************************

'**************************************
'MODULE/CLASS-WIDE CONSTANTS
'**************************************
'Strings...
Private Const m_sMODULE_NAME As String = "Dimensions"


'**************************************
'MODULE/CLASS-WIDE VARIABLES
'**************************************
'Collections...
Private m_coll As Collection
Private m_collKeys As Collection

'Enums...
Private m_enumCollScope As ProjDimenScope

'============================================================
'======================  EVENTS  ============================
'============================================================
'=============================================================================
Private Sub Class_Initialize()
Set m_coll = New Collection
Set m_collKeys = New Collection
End Sub
'=============================================================================
Private Sub Class_Terminate()
'Since there's really nothing useful an error-handler can do at this point...
```

```
Dimensions - 2

On Error Resume Next
Set m_coll = Nothing
Set m_collKeys = Nothing
End Sub
'=============================================================
'==================  (Private) FUNCTIONS  ===================
'=============================================================
'=========================================================================================
Private Function GetNumericIndexFromStringKey(ByVal Key) As Byte
Dim iKey As Byte
Dim i As Byte
'Get the integer key by looping through the keys collection...
For i = 1 To m_collKeys.Count
    If StrComp(Trim(Key), Trim(m_collKeys(i))) = 0 Then
        iKey = i
        i = m_collKeys.Count
    End If
Next i
GetNumericIndexFromStringKey = iKey
End Function




'*****************************************************************************************
'*****************************************************************************************
'*****************************  PUBLIC MEMBERS  ******************************************
'*****************************************************************************************
'*****************************************************************************************

'=============================================================
'======================  PROPERTIES  =========================
'=============================================================
'=========================================================================================
'Property CollectionScope...
Property Let CollectionScope(value As ProjDimenScope)
m_enumCollScope = value
End Property
Property Get CollectionScope() As ProjDimenScope
CollectionScope = m_enumCollScope
End Property
'Property NewEnum...
Public Property Get NewEnum() As IUnknown
        Set NewEnum = m_coll.[_NewEnum]
End Function

'=============================================================
'==================  (Public) FUNCTIONS  =====================
'=============================================================
'=========================================================================================
Public Function Items() As Collection
Set Items = m_coll
End Function
'=========================================================================================
Public Function Keys() As Collection
Set Keys = m_collKeys
End Function
'=========================================================================================
Public Function Item(index) As Dimension
If IsNumeric(index) Then
    index = Val(index) '...just in case it is passed in as a string
    Set Item = m_coll(index)
Else
    Set Item = m_coll(GetNumericIndexFromStringKey(index))
End If
End Function
'=========================================================================================
Public Function Count() As Integer
Count = m_coll.Count
End Function
'=========================================================================================
```

```
Dimensions - 3

Public Function ContainsItem( _
        ByVal ItemName As String, _
        ByVal CompMeth As VbCompareMethod) As Boolean
Dim bContains As Boolean

If m_coll.Count > 0 Then
        Dim Dimen As Dimension
        Dim i As Integer
        For i = 1 To m_coll.Count
                Set Dimen = m_coll.Item(i)
                If StrComp(Dimen.name, ItemName, CompMeth) = 0 Then
                        bContains = True
                        Exit For
                End If
        Next i
End If
ContainsItem = bContains
End Function




'============================================================
'=================  (Public) PROCEDURES  ==================
'============================================================
'============================================================================
Public Sub Add(VDimen As Dimension)
'No need for error handling; incompatible types will not even compile...
m_coll.Add Item:=VDimen, Key:=VDimen.name
m_collKeys.Add Item:=VDimen.name, Key:=VDimen.name
End Sub
'============================================================================
Public Sub Remove(index, ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "Remove()"
'Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

m_coll.Remove index
m_collKeys.Remove index

ExitPoint:
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'============================================================================
Public Sub RemoveAll()
Set m_coll = New Collection
Set m_collKeys = New Collection
End Sub
'============================================================================
Public Function ICloneable_Clone(ByRef RanOkRESULT As Boolean) As Object
```

```
Dimensions - 4

'Error-handling declarations...
Const sSOURCE As String = "PublFuncFullErrorHandler()"
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim clone As Dimensions
Dim Dimen As Dimension

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...

Set clone = New Dimensions
'Memberwise clone (i.e., properties)...
clone.CollectionScope = Me.CollectionScope

For Each Dimen In Me.Items
    clone.Add Dimen
Next Dimen

Set ICloneable_Clone = clone

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Function

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
            ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bFUNCTION_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Function
```

```
'.....................................................................................
'.....................................................................................
'.......        Developer:              William H. White (consultant)
'.......        With:                   TEKsystems Inc.
'.......                                www.teksystems.com
'.......        For:                    AIG
'.......                                Financial Information Systems
'.......                                1 NY Plaza, 15th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         8/20/2013
'.......        Proj finished:          11/21/2013
'.......        Proj revised:           1/15/2014
'.....................................................................................
'.....................................................................................



'*************************************************************************************
'*************************************************************************************
'*************************  CLASS-LEVEL DECLARATIONS  ************************
'*************************************************************************************
'*************************************************************************************
Option Explicit
Option Compare Binary '...use cAsE senSITive string comparisons




'*************************************************************************************
'*************************************************************************************
'*****************************  PRIVATE MEMBERS  *****************************
'*************************************************************************************
'*************************************************************************************

'***************************************
'MODULE/CLASS-WIDE CONSTANTS
'***************************************
'Strings...
Private Const m_sMODULE_NAME As String = "FileHandler"
Private Const m_sWSH_PWD As String = "StrongPwd123"
Private Const m_sBAD_WS_SUFFIX As String = "_BAD"

'Bytes...
Private Const m_yCART_META_COLS As Byte = 3
Private Const m_yCART_WSHTYPE_COL As Byte = 1
Private Const m_yCART_WSH_COL As Byte = 2
Private Const m_yCART_CELL_COL As Byte = 3
Private Const m_yDIMENS_NON_AMT As Byte = 25
Private Const m_yDIMENS_AMT As Byte = 4


'These columns are WITHOUT including Meta Data columns!!!....
Private Const m_yCOL_DIMEN_TRANS_AMT As Byte = 26
Private Const m_yCOL_DIMEN_LOC_AMT As Byte = 27
Private Const m_yCOL_DIMEN_FUNC_AMT As Byte = 28
Private Const m_yCOL_DIMEN_USD_AMT As Byte = 29

'***************************************
'MODULE/CLASS-WIDE TYPES
'***************************************
Private Type m_TypDimNmVal
    nm As String
    SLRCol As Byte
    Val As String
End Type

Private Type m_TypDimNmRC
```

```
FileHandler - 2

    nm As String
    SLRCol As Byte
    RowCol As Long
End Type

Private Type m_CARTColVal
    Col As Byte
    Val As String
End Type

'**************************************
'MODULE/CLASS-WIDE VARIABLES
'**************************************




'============================================================
'======================  EVENTS  ============================
'============================================================
'=============================================================================
Private Sub Class_Initialize()
'[VBA QUIRK:  Errors in Intialize, Activate, and Terminate event handlers _
cannot be handled by procedures which create or destroy instances of the _
class. If any such error-handling IS required, create a separate public _
procedure - e.g., the Initialize procedure, below.]
End Sub
'=============================================================================
Private Sub Class_Terminate()
'[VBA QUIRK:  Errors in Intialize, Activate, and Terminate event handlers _
cannot be handled by procedures which create or destroy instances of the _
class. If any such error-handling IS required, create a separate public _
procedure - e.g., the Initialize procedure, below.]

'Since there's really nothing useful an error-handler can do at this point...
On Error Resume Next
'clean-up code...
End Sub



'============================================================
'==================  (Private) FUNCTIONS  ==================
'============================================================
'=============================================================================
Private Function GetValueInMergedCells( _
    ByRef MergedCell As Range, _
    ByVal Scope As ProjDimenScope, _
    ByRef RanOkRESULT As Boolean) As String
'Error-handling declarations...
Const sSOURCE As String = "GetValueInMergedCells()"
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim rngMergeArea As Range
Dim c As Range
Dim sVal As String
Dim yMergeAreaSize As Byte
Dim i As Byte

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Set rngMergeArea = MergedCell.MergeArea
Select Case Scope
    Case projScopeCol
        yMergeAreaSize = rngMergeArea.Columns.Count
        For i = 1 To yMergeAreaSize
            sVal = Trim(rngMergeArea.Cells(1, i))
```

```
FileHandler - 3

            If StrComp(sVal, vbNullString) <> 0 Then i = yMergeAreaSize '...break loop
        Next i
    Case projScopeRow
        yMergeAreaSize = rngMergeArea.Rows.Count
        For i = 1 To yMergeAreaSize
            sVal = Trim(rngMergeArea.Cells(i, 1))
            If StrComp(sVal, vbNullString) <> 0 Then i = yMergeAreaSize '...break loop
        Next i
    Case Else
        Err.Raise g_lERR_INAPPROP_DIMEN_SCOPE
End Select

GetValueInMergedCells = sVal

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Function

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
            ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bFUNCTION_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Function
'================================================================================
Private Function DoesSourceWbkHaveOur2DimenWshs( _
    ByRef DimenLocWshsMissingRESULT As Boolean, _
    ByRef ValListsWshMissingRESULT As Boolean, _
    ByRef RanOkRESULT As Boolean) As Boolean
'Error-handling declarations...
Const sSOURCE As String = "DoesSourceWbkHaveOur2DimenWshs()"
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
'''Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim ws As Worksheet
Dim bHasDimenLocWsh As Boolean
Dim bHasValidLstsWsh As Boolean

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
For Each ws In g_wbSrcData.Worksheets
    If StrComp(ws.name, g_sWSH_DIMEN_LOCS) = 0 Then bHasDimenLocWsh = True
    If StrComp(ws.name, g_sWSH_VAL_LISTS) = 0 Then bHasValidLstsWsh = True
    If bHasDimenLocWsh And bHasValidLstsWsh Then Exit For
Next ws

'Trap for user having somehow deleted or renamed one of our worksheets...
If bHasDimenLocWsh And Not bHasValidLstsWsh Then ValListsWshMissingRESULT = True
If bHasValidLstsWsh And Not bHasDimenLocWsh Then _
    DimenLocWshsMissingRESULT = True

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
DoesSourceWbkHaveOur2DimenWshs = bHasDimenLocWsh And bHasValidLstsWsh
Exit Function
```

```
FileHandler - 4

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bFUNCTION_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Function



'============================================================
'=================  (Private) PROCEDURES  ================
'============================================================
'============================================================================
Private Sub OpenSourceFile( _
        ByRef FileSelectedRESULT As Boolean, _
        ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "OpenSourceFile()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim fd As FileDialog
'Even though the returned selection is a string, the FOR EACH loop _
in .SelectedItems returns the string as a variant...
Dim vrtSelection As Variant

'On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

Set fd = Application.FileDialog(msoFileDialogOpen)

'If g_bStepThruMode Then Stop

With fd
    .AllowMultiSelect = False
    .ButtonName = "Open File"
    .Filters.Add "Excel Files", "*.xlsx; *.xlsm; *.xls; *.xlm; *.xlw", 1
    .Filters.Add "Text Files", "*.prn; *.txt; *.csv", 2
    .FilterIndex = 1
    .Title = "STEP 1 OF 8:  Open Source Data"
    'Show returns -1 if the user selects a file, 0 if the user cancels...
    If .Show = -1 Then
        'Step through each string in the FileDialogSelectedItems collection.
        For Each vrtSelection In .SelectedItems
        '.Item
            'sSelection is a String that contains the path of each selected item.
            'You can use any file I/O functions that you want to work with this path.
            'This example simply displays the path in a message box.
            g_sSrcWbkFullNm = vrtSelection
            FileSelectedRESULT = True
        Next vrtSelection
    End If
    .Execute
End With

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub
```

```
FileHandler - 5

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Private Sub SetSourceWbkPlusNmAndPath(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "SetSourceWbkPlusNmAndPath()"
Const bSUB_IS_ENTRY_PT As Boolean = False
'''Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim iLastSlash As Integer
'NOTE:  chr(92) = "\"

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'If g_bStepThruMode Then Stop
'CODE HERE...
iLastSlash = InStrRev(g_sSrcWbkFullNm, Chr(92))
g_sSrcWbkPath = Left(g_sSrcWbkFullNm, iLastSlash)
g_sSrcWbkNm = Right(g_sSrcWbkFullNm, Len(g_sSrcWbkFullNm) - iLastSlash)
Set g_wbSrcData = Workbooks(g_sSrcWbkNm)

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Private Sub RemoveOldXLNmzAndAdd2WshsToSourceWbk(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "RemoveOldXLNmzAndAdd2WshsToSourceWbk()"
Const bSUB_IS_ENTRY_PT As Boolean = False
'''Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Const yXL_NMZ_TO_CHK As Byte = 7
Dim nm As name
Dim sXLNmzToChk(0 To yXL_NMZ_TO_CHK - 1) As String
Dim i As Byte
Dim j As Integer
Dim iSourceWbkWshs As Integer
```

```
FileHandler - 6

Dim iNmbrExistingNmz As Integer
Dim vsbOrigSLDimLocs As XlSheetVisibility
Dim vsbOrigSLValLists As XlSheetVisibility

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
If g_bStepThruMode Then Stop
If Not g_bDebugMode Then Application.ScreenUpdating = False

'Handle wsh visibility...
vsbOrigSLDimLocs = WshSLDimLocs.Visible
vsbOrigSLValLists = WshSLValLists.Visible
WshSLDimLocs.Visible = xlSheetVisible
WshSLValLists.Visible = xlSheetVisible

'We tend to have certain range names which linger in source data workbooks _
in cases where the workbook is opened by CART, but then closed with the _
two added worksheets deleted.  This is essentially a development- and _
testing-only issue.  Still, to resolve it we'll look for and clean out those _
range names...
If iNmbrExistingNmz > 0 Then
    sXLNmzToChk(0) = "valcolWshNm"
    sXLNmzToChk(1) = "valcolWshsToImport"
    sXLNmzToChk(2) = "valcolWshTypes"
    sXLNmzToChk(3) = "valhdrWshsToImport"
    sXLNmzToChk(4) = "valhdrWshTypes"
    sXLNmzToChk(5) = "valhdrzWshNmTypeMap"
    sXLNmzToChk(6) = "valhdrzWshTypeInfo"

    If g_bStepThruMode Then Stop
    For i = 0 To yXL_NMZ_TO_CHK - 1
        'Reset this after each "i" in case we have, in fact, deleted a name...
        iNmbrExistingNmz = g_wbSrcData.Names.Count
        For j = 1 To iNmbrExistingNmz
            'Work backwards, since we are deleting data...
            Set nm = g_wbSrcData.Names(1 + iNmbrExistingNmz - j)
            If StrComp(Trim(nm.name), Trim(sXLNmzToChk(i)), vbTextCompare) = 0 Then _
                g_wbSrcData.Names(1 + iNmbrExistingNmz - j).Delete
        Next j
    Next i
End If

'Copy our 2 worksheets into the source data workbook...
iSourceWbkWshs = g_wbSrcData.Worksheets.Count
WshSLDimLocs.Copy after:=g_wbSrcData.Worksheets(iSourceWbkWshs)
WshSLValLists.Copy after:=g_wbSrcData.Worksheets(iSourceWbkWshs + 1)


'Restore Wsh visibility...
WshSLDimLocs.Visible = vsbOrigSLDimLocs
WshSLValLists.Visible = vsbOrigSLValLists

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Application.ScreenUpdating = True
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
```

```
FileHandler - 7

                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Private Sub SetOur2SrcWbkWshsAndPwdProtect(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "SetOur2SrcWbkWshsAndPwdProtect()"
Const bSUB_IS_ENTRY_PT As Boolean = False
'''Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wsDimenLocs As Worksheet
Dim wsValLists As Worksheet


On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'Set worksheets...
Set wsDimenLocs = g_wbSrcData.Worksheets(g_sWSH_DIMEN_LOCS)
Set wsValLists = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)

'Password protect...
If g_bDebugMode Then
    wsValLists.Unprotect g_sWSH_PASSWORD
    wsDimenLocs.Unprotect g_sWSH_PASSWORD
Else
    wsValLists.Protect Password:=g_sWSH_PASSWORD, contents:=True, userinterfaceonly:=True
    wsDimenLocs.Protect Password:=g_sWSH_PASSWORD, contents:=True, userinterfaceonly:=True
End If

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Private Sub RemoveRemainingValWshFromSourceWhenOtherMissing( _
    ByVal TypeMissingWsh As ProjMissingWsh, _
    ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "RemoveRemainingValWshFromSourceWhenOtherMissing()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wsRemaining As Worksheet
Dim sMissingWsh As String
Dim sFoundWsh As String

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True
```

```
FileHandler - 8


'CODE HERE...
Select Case TypeMissingWsh
    Case ProjMissingWsh.projMissingDimenLocs
        sMissingWsh = g_sWSH_DIMEN_LOCS
        sFoundWsh = g_sWSH_VAL_LISTS
    Case ProjMissingWsh.projMissingValLists
        sMissingWsh = g_sWSH_VAL_LISTS
        sFoundWsh = g_sWSH_DIMEN_LOCS
End Select

'Notify user...
g_cMsgBoxHandler.SetYouLostOneOfOurWshsMsg _
    WshMissing:=sMissingWsh, _
    WshFound:=sFoundWsh
g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbExclamation

'Delete remaining worksheet...
Set wsRemaining = g_wbSrcData.Worksheets(sFoundWsh)
'Suppress prompt about worsheet possibly containing data...
Application.DisplayAlerts = False
wsRemaining.Delete

ExitPoint:
On Error Resume Next
Application.DisplayAlerts = True
ThisWorkbook.Activate
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Private Sub ArrangeWbkWindows()
Dim win As Window

'We ran into problems here during Leo's first demo (ugh!).  His machine _
has some hidden file called EcoLab.xls, and the code through an error _
trying to minimize that Window.  Ergo, we are just going to...
On Error Resume Next
For Each win In Windows
    If StrComp(win.Parent.name, ThisWorkbook.name) <> 0 And _
        StrComp(win.Parent.name, g_sSrcWbkNm) <> 0 Then _
            win.WindowState = xlMinimized
Next win
Windows.Arrange xlArrangeStyleCascade
End Sub
'================================================================================
Private Sub CheckForPriorAbortedWizardProcessing( _
        ByRef MaxNextStageRESULT As ProjNextStage, _
        ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "CheckForPriorAbortedWizardProcessing()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wsValLists As Worksheet
```

```
FileHandler - 9

Dim rngDataWshs As Range
Dim rngWshTypeStatusTbl As Range
Dim rngWshTypeWshNmMapTbl As Range
Dim bFileDimensDefined As Boolean
Dim bWshTypeDimensDefined As Boolean

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Set wsValLists = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
Set rngDataWshs = wsValLists.Range("vallstdodynWshsToImport")
Set rngWshTypeStatusTbl = wsValLists.Range("valtbldodynWshTypeInfo")
Set rngWshTypeWshNmMapTbl = wsValLists.Range("valtbldodynWshNmTypeMap")

'NOTE:  We work 'backwards' in this procedure, from most advanced to least _
advanced to see where we should set the next stage enum...

'If g_bStepThruMode Then Stop
'See if any worksheet type dimensions have been defined...
bWshTypeDimensDefined = g_cDimHandler.AnyWshTypeDimensDefined( _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
If bWshTypeDimensDefined Then
    MaxNextStageRESULT = projNxtStgWshTypeDimens
    GoTo ExitPoint
Else
    MaxNextStageRESULT = projNxtStgWshTypeDimens
End If

'See if the user set up any worksheet type/worksheet name mappings...
If StrComp(Trim(rngWshTypeWshNmMapTbl.Cells(1, g_yWSH_ASSOC_TBL_WSH_NM_COL)), _
        vbNullString) = 0 Then MaxNextStageRESULT = projNxtStgAssocWshTyps

'We can't test for whether the user got as far as creating Wsh Types, because _
we always make sure to retain at least one Wsh Type (the ALL type) in _
our data workbooks. HOWEVER, if we show ONLY one WshType then we want to _
disable the Associate Wsh Types button...
If rngWshTypeStatusTbl.Rows.Count = 1 Then
    MaxNextStageRESULT = projNxtStgCreateWshTyps
    RibbonHandler.SetOnly1WshTypeFlag True
End If

'See if the user specified any data worksheets...
If StrComp(Trim(rngDataWshs.Cells(1, 1)), vbNullString) = 0 Then _
        MaxNextStageRESULT = projNxtStgIDDataWshs

'NOTE:  We won't test for whether the user defined any file dimensions, _
because it is quite plausible that a user will define everything at the _
worksheet type level.


ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
            ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
```

```
FileHandler - 10

        Resume ExitPoint
    End If
End Sub
'================================================================================
Private Sub HandleHiddenRenamedOrDeletedDataWshs( _
    ByRef DataWshsHiddenRESULT As Boolean, _
    ByRef UserAnsHiddenWshsRESULT As VbMsgBoxResult, _
    ByRef DataWshsRenamedOrDeletedRESULT As Boolean, _
    ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "HandleHiddenRenamedOrDeletedDataWshs()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wsValLists As Worksheet
Dim ws As Worksheet
Dim rngDataWshsDataOnly As Range
Dim rngDataWshsWithHdrz As Range
Dim c As Range
Dim saHiddenWshs() As String
Dim saInvalidWshs() As String
Dim iCurrentSrcWbkWshs As Integer
Dim yHiddenWshs As Byte
Dim yInvalidWshs As Byte
Dim i As Byte
Dim j As Byte
Dim bWshHidden As Boolean
Dim bWshStillExists As Boolean

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Set wsValLists = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
Set rngDataWshsDataOnly = wsValLists.Range("vallstdodynWshsToImport")
Set rngDataWshsWithHdrz = rngDataWshsDataOnly.Offset(-1, 0)
Set rngDataWshsWithHdrz = _
        rngDataWshsWithHdrz.Resize(rngDataWshsDataOnly.Rows.Count + 1, 1)
iCurrentSrcWbkWshs = g_wbSrcData.Worksheets.Count

'Just in case...
If rngDataWshsDataOnly.Rows.Count = 0 Then
    DataWshsRenamedOrDeletedRESULT = False
    GoTo ExitPoint
End If

'Loop through each previously-defined data worksheet to see if it _
still exists and/or is hidden...
For Each c In rngDataWshsDataOnly
    bWshHidden = False '...reset
    For i = 1 To iCurrentSrcWbkWshs
        Set ws = g_wbSrcData.Worksheets(i)
        'If one of our data wshs see if hidden...
        If StrComp(Trim(c), ws.name, vbTextCompare) = 0 And _
                ws.Visible <> xlSheetVisible Then
            bWshHidden = True
            i = iCurrentSrcWbkWshs '...break loop
        End If
        If bWshHidden Then
            ReDim Preserve saHiddenWshs(0 To yHiddenWshs)
            saHiddenWshs(yHiddenWshs) = Trim(c)
            yHiddenWshs = yHiddenWshs + 1
        End If
    Next i
Next c

If g_bStepThruMode Then Stop
'Ask user about hidden wshs, process those based on answer...
If yHiddenWshs > 0 Then
    g_cMsgBoxHandler.AskUnhideDataWshsQuest saHiddenWshs
```

```
    UserAnsHiddenWshsRESULT = _
            g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbYesNoCancel + vbQuestion)
    DataWshsHiddenRESULT = True
    Select Case UserAnsHiddenWshsRESULT
        Case vbYes
            For i = 1 To yHiddenWshs
                Set ws = g_wbSrcData.Worksheets(saHiddenWshs(i - 1))
                ws.Visible = xlSheetVisible
            Next i
        Case vbNo
            'Remove hidden wshs from list of data wshs...
            For Each c In rngDataWshsDataOnly
                For i = 1 To yHiddenWshs
                    If Trim(c) = saHiddenWshs(i - 1) Then c.ClearContents
                Next i
            Next c
            'Sort to close up cleared cells, then reset range def...
            rngDataWshsDataOnly.Sort _
                    key1:=rngDataWshsDataOnly.Columns(1), _
                    Header:=xlNo, _
                    Orientation:=xlSortColumns
            Set rngDataWshsDataOnly = _
                    wsValLists.Range("vallstdodynWshsToImport")
            'Remove wsh from name/type mapping table...
            Dim rngTblWshNmTypeMpg As Range
            Dim sWshType As String
            Dim yWshNmTypeRows As Byte
            Set rngTblWshNmTypeMpg = wsValLists.Range("valtbldodynWshNmTypeMap")
            yWshNmTypeRows = rngTblWshNmTypeMpg.Rows.Count
            For i = 1 To yWshNmTypeRows
                For j = 1 To yHiddenWshs
                    If StrComp(Trim(rngTblWshNmTypeMpg.Cells( _
                            i, g_yWSH_ASSOC_TBL_WSH_NM_COL)), _
                            saHiddenWshs(j - 1), vbTextCompare) = 0 Then
                        rngTblWshNmTypeMpg.Cells( _
                                i, g_yWSH_ASSOC_TBL_WSH_NM_COL).ClearContents
                        rngTblWshNmTypeMpg.Cells( _
                                i, g_yWSH_ASSOC_TBL_TYPE_COL).ClearContents
                        j = yHiddenWshs
                    End If
                Next j
            Next i
            'Sort to close up cleared cells, then reset range def...
            rngTblWshNmTypeMpg.Sort _
                    key1:=rngTblWshNmTypeMpg.Columns(g_yWSH_ASSOC_TBL_TYPE_COL), _
                    key2:=rngTblWshNmTypeMpg.Columns(g_yWSH_ASSOC_TBL_WSH_NM_COL), _
                    Header:=xlNo, _
                    Orientation:=xlSortColumns
            Set rngTblWshNmTypeMpg = wsValLists.Range("valtbldodynWshNmTypeMap")
            'Remove any empty wsh types...
            Dim wsDimenLocs As Worksheet
            Dim rngTblWshTypes As Range
            Dim rngEmptyDimLocCol As Range
            Dim yWshTypes As Byte
            Dim bEmptyWshType As Boolean
            Set wsDimenLocs = g_wbSrcData.Worksheets(g_sWSH_DIMEN_LOCS)

            Set rngTblWshTypes = wsValLists.Range("valtbldodynWshTypeInfo")
            yWshTypes = rngTblWshTypes.Rows.Count
            'Since we have deleted cells we must recalc...
            yWshNmTypeRows = rngTblWshNmTypeMpg.Rows.Count
            For i = 1 To yWshTypes
                bEmptyWshType = True '...reset
                sWshType = _
                        Trim(rngTblWshTypes.Cells(i, g_yWSH_TYPE_TBL_NM_COL))
                For j = 1 To yWshNmTypeRows
                    If StrComp(sWshType, _
                            Trim(rngTblWshNmTypeMpg.Cells( _
                                    j, g_yWSH_ASSOC_TBL_TYPE_COL)), _
                                    vbTextCompare) = 0 Then
                        bEmptyWshType = False
```

```
                    j = yWshNmTypeRows '...break FOR loop
                End If
            Next j
            If bEmptyWshType Then
                rngTblWshTypes.Cells(i, _
                        g_yWSH_TYPE_TBL_NM_COL).ClearContents
                rngTblWshTypes.Cells(i, _
                        g_yWSH_TYPE_TBL_DEFINED_COL).ClearContents
                Set rngEmptyDimLocCol = _
                        wsDimenLocs.Range("tblDimenLocs" & _
                        Chr(91) & sWshType & Chr(93))
                rngEmptyDimLocCol.EntireColumn.Delete xlShiftToLeft
            End If
        Next i
        'Sort to close up any empty rows...
        rngTblWshTypes.Sort _
            key1:=rngTblWshTypes.Columns(g_yWSH_TYPE_TBL_NM_COL), _
            Header:=xlNo, _
            Orientation:=xlSortColumns
        Set rngTblWshTypes = wsValLists.Range("valtbldodynWshTypeInfo")
        'In case we wiped out the one and only WshType...
        If IsEmpty(rngTblWshTypes.Cells(1, g_yWSH_TYPE_TBL_NM_COL)) Then
            rngTblWshTypes.Cells(1, g_yWSH_TYPE_TBL_NM_COL) = _
                    g_sWSH_TYPE_ALL_PC
            rngTblWshTypes.Cells(1, g_yWSH_TYPE_TBL_DEFINED_COL) = False
        End If
    Case vbCancel
        GoTo ExitPoint
    End Select
End If

'Loop through each previously-defined data worksheet to see if it _
still exists...
For Each c In rngDataWshsDataOnly
    bWshStillExists = False '...reset
    'Check against each ws in data wbk...
    For i = 1 To iCurrentSrcWbkWshs
        Set ws = g_wbSrcData.Worksheets(i)
        If StrComp(Trim(c), ws.name, vbTextCompare) = 0 Then
            bWshStillExists = True
            i = iCurrentSrcWbkWshs '...break loop
        End If
    Next i
    If Not bWshStillExists Then
        yInvalidWshs = yInvalidWshs + 1
        ReDim Preserve saInvalidWshs(0 To yInvalidWshs - 1)
        saInvalidWshs(yInvalidWshs - 1) = Trim(c)
        c.ClearContents
    End If
Next c

'Now tell user about invalid wshs...
If yInvalidWshs > 0 Then
    'Close up the empty spaces in our list of data worksheets (NOTE:  sorting _
    on the data-only range does NOT push the empty cells to the bottom)...
    rngDataWshsWithHdrz.Sort _
            key1:=rngDataWshsDataOnly.Columns(1), _
            Header:=xlYes, Orientation:=xlSortColumns

    'Remove invalid worksheets from Wshtype/WshName Mapping table...
    Dim rngWshTypeWshNmTbl As Range
    Dim rngRowWshTypeWshNmTbl As Range
    Dim iDataWshs As Integer
    Set rngWshTypeWshNmTbl = wsValLists.Range("valtbldodynWshNmTypeMap")
    iDataWshs = rngWshTypeWshNmTbl.Rows.Count
    For i = 1 To iDataWshs
        'See if the wsh name is in our invalid list...
        For j = 1 To yInvalidWshs
            If StrComp(Trim(rngWshTypeWshNmTbl.Cells(i, _
                    g_yWSH_ASSOC_TBL_WSH_NM_COL)), _
                    saInvalidWshs(j - 1), _
```

```
                    vbTextCompare) = 0 Then
                Set rngRowWshTypeWshNmTbl = rngWshTypeWshNmTbl.Rows(i)
                rngRowWshTypeWshNmTbl.ClearContents
                j = yInvalidWshs '...break loop
            End If
        Next j
    Next i
    'Close up empty spaces in our list of data worksheets...
    rngWshTypeWshNmTbl.Sort _
            key1:=rngWshTypeWshNmTbl.Columns(g_yWSH_ASSOC_TBL_TYPE_COL), _
            key2:=rngWshTypeWshNmTbl.Columns(g_yWSH_ASSOC_TBL_WSH_NM_COL), _
            Header:=xlNo
    'Save changes, set our output flag, and notify user...
    g_wbSrcData.Save
    DataWshsRenamedOrDeletedRESULT = True
    g_cMsgBoxHandler.SetDeletedOrRenamedWshsMsg yInvalidWshs, saInvalidWshs()
    g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbExclamation
End If


ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
            ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'================================================================================
Private Sub PopulateFileDimensionsAndValues(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "PopulateFileDimensionsAndValues()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wsDimenTbl As Worksheet
Dim tblDimenLocs As ListObject
Dim rngDimenTbl As Range
Dim rngDimenTblDimenNmz As Range
Dim c As Range
Dim sRefAddress As String
Dim sDimLocTblWshTypeColAddr As String
Dim yDimenTblWshTypeCol As Byte
Dim yColOffsetNmToWshType As Byte
Dim bGetWshTypeDimen As Boolean
Dim bRngRefDefined As Boolean

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Set wsDimenTbl = g_wbSrcData.Worksheets(g_sWSH_DIMEN_LOCS)


ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
```

```
FileHandler - 14

Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Private Sub TrapForAndPromptReProtectedWshs( _
        ByRef WshsProtectedRESULT As Boolean, _
        ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "PublSubFullErrorHandler()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim collProtectedWshs As Collection
Dim wsValLists As Worksheet
Dim wsSrc As Worksheet
Dim rngDataWshs As Range
Dim c As Range

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Set collProtectedWshs = New Collection
Set wsValLists = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
Set rngDataWshs = wsValLists.Range("vallstdodynWshsToImport")
For Each c In rngDataWshs
    Set wsSrc = g_wbSrcData.Worksheets(c.value)
    If wsSrc.ProtectContents Then _
            collProtectedWshs.Add Item:=wsSrc, Key:=wsSrc.name
Next c

If collProtectedWshs.Count > 0 Then
    WshsProtectedRESULT = True
    g_cMsgBoxHandler.SetProtectedWshsMsg collProtectedWshs
    g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbExclamation
End If

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
            ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
```

```
FileHandler - 15

        Resume ExitPoint
    End If
End Sub
'===============================================================================
Private Sub RemoveImportedDataAndDeleteBADWshs( _
        ByVal LastDataRow As Long, _
        ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "RemoveImportedDataAndDeleteBADWshs()"
Const bSUB_IS_ENTRY_PT As Boolean = False
'Operating code declarations...
Const yCART_WSH_HDR_ROWS As Byte = 1
Dim ws As Worksheet
Dim rngRowsToDelete As Range
Dim lDataRowFirst As Long
Dim lDataColFirst As Long

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Set rngRowsToDelete = _
        WshGoodData.Range(WshGoodData.Cells(yCART_WSH_HDR_ROWS + 1, 1), _
        WshGoodData.Cells(LastDataRow, 1))
rngRowsToDelete.EntireRow.Delete xlShiftUp

'So the user doesn't get the "This worksheet may contain data..." prompt...
Application.DisplayAlerts = False
For Each ws In ThisWorkbook.Worksheets
    If StrComp(Right(ws.name, Len(m_sBAD_WS_SUFFIX)), _
            m_sBAD_WS_SUFFIX) = 0 Then ws.Delete
Next ws
Application.DisplayAlerts = True '...restore matters
ThisWorkbook.Activate
WshGoodData.Select
'Do a "ctrl-Home"...
Cells(ActiveWindow.SplitRow + 1, ActiveWindow.SplitColumn + 1).Select

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
            ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub



'*******************************************************************************
'*******************************************************************************
'*****************************  PUBLIC MEMBERS  ********************************
'*******************************************************************************
'*******************************************************************************

'=========================================================
'======================  PROPERTIES  ======================
```

```
'============================================================
'============================================================================

'============================================================
'==================  (Public) FUNCTIONS  ===================
'============================================================
'============================================================================
Public Function IsSourceWbkStillOpen(ByRef RanOkRESULT As Boolean) As Boolean
'Error-handling declarations...
Const sSOURCE As String = "IsSourceWbkStillOpen()"
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wb As Workbook
Dim bWbkOpen As Boolean '...False

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

For Each wb In Workbooks
    If StrComp(wb.FullName, g_sSrcWbkFullNm) = 0 Then
        bWbkOpen = True
        Exit For
    End If
Next wb

IsSourceWbkStillOpen = bWbkOpen

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Function

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bFUNCTION_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Function




'============================================================
'==================  (Public) PROCEDURES  ==================
'============================================================
'============================================================================
Public Sub OpenSourceWbkMaestro(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "OpenSourceWbkMaestro()"
Dim bCalledProcedureRanOk As Boolean
'''Operating code declarations...
Dim wsOrigActive As Worksheet
Dim enumNextStg As ProjNextStage
Dim ansUnhideHiddenWshs As VbMsgBoxResult
Dim bSrcFileOpened As Boolean
Dim bMissingDimenLocWsh As Boolean
Dim bMissingValListsWsh As Boolean
```

```
FileHandler - 17

Dim bWshsHidden As Boolean
Dim bAllMandDimensDefined As Boolean

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'OPERATING CODE HERE...
'If g_bStepThruMode Then Stop
OpenSourceFile _
        FileSelectedRESULT:=bSrcFileOpened, _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
If Not bSrcFileOpened Then GoTo ExitPoint

'If g_bStepThruMode Then Stop
SetSourceWbkPlusNmAndPath RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
'If g_bStepThruMode Then Stop
Set wsOrigActive = g_wbSrcData.ActiveSheet
If Not DoesSourceWbkHaveOur2DimenWshs( _
        DimenLocWshsMissingRESULT:=bMissingDimenLocWsh, _
        ValListsWshMissingRESULT:=bMissingValListsWsh, _
        RanOkRESULT:=bCalledProcedureRanOk) Then
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    'Trap for source workbook having one of our worksheets but not the other...
    'If g_bStepThruMode Then Stop
    If bMissingDimenLocWsh Then
        RemoveRemainingValWshFromSourceWhenOtherMissing _
                TypeMissingWsh:=projMissingDimenLocs, _
                RanOkRESULT:=bCalledProcedureRanOk
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    End If
    If bMissingValListsWsh Then
        RemoveRemainingValWshFromSourceWhenOtherMissing _
                TypeMissingWsh:=projMissingValLists, _
                RanOkRESULT:=bCalledProcedureRanOk
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    End If
    'Skip the prompt if the user opened a damaged source wbk and we _
    just told him about how we had to remove his one remaining _
    worksheet of ours...
    If Not bMissingDimenLocWsh And Not bMissingValListsWsh Then
        g_cMsgBoxHandler.SetWillAddSLWshsToSrcWbkMsg _
            WshNm1:=g_sWSH_DIMEN_LOCS, _
            WshNm2:=g_sWSH_VAL_LISTS
        g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbInformation
    End If
    RemoveOldXLNmzAndAdd2WshsToSourceWbk RanOkRESULT:=bCalledProcedureRanOk
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    g_bWizardMode = True
    enumNextStg = projNxtStgFileDimens
Else
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
End If

'If g_bStepThruMode Then Stop
SetOur2SrcWbkWshsAndPwdProtect RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
wsOrigActive.Activate '...restore matters
ArrangeWbkWindows '...no error handling
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

'The object factory is instantiated on WbkOpen.  We then want to _
create our other global objects now, especially for a user re-opening _
a source wbk...
g_cObjFactory.CreateGlobalDimensionHandler RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

'If g_bStepThruMode Then Stop
'If this is a previously-opened workbook see if the user finished the entire _
```

```
wizard process.  If the user didn't identify how far the user got and _
set our ribbon accordingly.  [NOTE:  This procedured only sets the next-stage _
module-level variable.  It does NOT fire the ResetRibbon procedure.]...
If Not g_bWizardMode Then
    CheckForPriorAbortedWizardProcessing _
            MaxNextStageRESULT:=enumNextStg, _
            RanOkRESULT:=bCalledProcedureRanOk
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
End If

If g_bStepThruMode Then Stop
'If appropriate check for possible renamed or deleted sheets...
If enumNextStg > projNxtStgIDDataWshs Then
    HandleHiddenRenamedOrDeletedDataWshs _
            DataWshsHiddenRESULT:=bWshsHidden, _
            UserAnsHiddenWshsRESULT:=ansUnhideHiddenWshs, _
            DataWshsRenamedOrDeletedRESULT:=g_bUserRenamedOrDeletedDataWshs, _
            RanOkRESULT:=bCalledProcedureRanOk
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
End If

'Finish handling hidden wshs...
If bWshsHidden Then
    Select Case ansUnhideHiddenWshs
        Case vbYes
            g_wbSrcData.Save
        Case vbNo
            g_wbSrcData.Save
        Case vbCancel
            g_cMsgBoxHandler.SetDecideOnHiddenWshsMsg
            g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbExclamation
            g_wbSrcData.Close savechanges:=False
            RibbonHandler.SetNextStage projNxtStgOpenSrc
            RibbonHandler.ResetRibbon
            GoTo ExitPoint
    End Select
End If

'Create our other global object, now that we know that status of NextStage...
g_cObjFactory.CreateGlobalFormsHandler _
        NextStage:=enumNextStg, _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

g_cDimHandler.PopulateWshTypesCollFromSrcWbkTbl _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

'Set ribbon module for single worksheet type...
If g_cDimHandler.WorksheetTypes.Count < 2 Then RibbonHandler.SetOnly1WshTypeFlag True

'If g_bStepThruMode Then Stop
'Final checks on which buttons to activate in the ribbon...
If Not g_bWizardMode Then
    ThisWorkbook.Activate
    'If appropriate see if mandatory dimens have been defined for all _
    worksheet types...
    If enumNextStg > projNxtStgAssocWshTyps Then
        bAllMandDimensDefined = _
            Not g_cDimHandler.AnyWshTypesWithUndefinedMandDimens( _
                    RanOkRESULT:=bCalledProcedureRanOk)
                If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    End If
    'Set ribbon button...
    If g_bUserRenamedOrDeletedDataWshs Then
        enumNextStg = projNxtStgIDDataWshs
    Else
        If bAllMandDimensDefined Then enumNextStg = projNxtStgImportData
    End If
    RibbonHandler.SetNextStage enumNextStg
    RibbonHandler.ResetRibbon
```

```
FileHandler - 19

    'If g_bStepThruMode Then Stop
Else
    If g_bStepThruMode Then Stop
    RibbonHandler.SetNextStage enumNextStg
    RibbonHandler.ResetRibbon
    If g_bStepThruMode Then Stop
    RibbonHandler.SetDimensionsMaestro _
        DimScope:=projScopeFile, _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
End If

ExitPoint:
On Error Resume Next
'''Must-run procedure/function clean-up code...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Public Sub HandleUserClosingSourceWbk(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "HandleUserClosingSourceWbk()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
g_sSrcWbkFullNm = vbNullString
g_sSrcWbkNm = vbNullString
g_sSrcWbkPath = vbNullString
Set g_wbSrcData = Nothing

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
```

```
FileHandler - 20

End Sub
'================================================================================
Public Sub ImportDataMaestro(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "ImportDataMaestro()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Const sCART_FILE_SUFFIX As String = "_CART_SLR_"
'From what I find on the intenet maximum wsh name length for XL '07 is 31. _
However, in practice Chirag Mehta has found that they are still getting errors, _
so I am knocking this down to 30 (whw 11/8/13)...
Const yXL_MAX_WSH_NM As Byte = 30 '...from what I find on the internet
Const yDIGITS_IN_BAD_WSH_NM_COUNTER = 2 '...i.e., "01", "02", etc.
'Dictionaries...
Dim dictBadHiddenOrEmptyRows As Dictionary
Dim dictBadHiddenOrEmptyCols As Dictionary
'DimenName/SLRCol/DimValue types...
Dim tFileDimenVals() As m_TypDimNmVal
'Worksheets...
Dim wsDimenLocs As Worksheet
Dim wsSrcData As Worksheet
Dim wsValLists As Worksheet
Dim wsBADData As Worksheet
'Ranges...
Dim rngValLstDataWshs As Range
Dim rngValLstWshTypes As Range
Dim rngValTblWshNmTypeMap As Range
Dim rngLocListDimNms As Range
Dim rngLocListSLRCols As Range
Dim rngLocListFileHC As Range
Dim rngLocListFileRR As Range
Dim rngLocListCurrWshType As Range
'Double...
Dim dRawAmt As Double
'Currency...
Dim cAdjAmt As Currency
'Longs...
Dim lScale As Long
Dim lBadHdrRow As Long
Dim lRowDimenCol As Long
Dim lColDimenRow As Long
Dim lDataRowFirst As Long
Dim lDataRowLastPrelim As Long
Dim lDataRowLast As Long
Dim lDataColFirst As Long
Dim lDataColLastPrelim As Long
Dim lDataColLast As Long
Dim lDataRows As Long
Dim lDataCols As Long
Dim lRowNo As Long
Dim lColNo As Long
Dim x As Long
Dim y As Long
Dim lCARTDataRow As Long
Dim lCARTWshRows As Long
Dim lCARTDataCol As Long
Dim lCARTRowNext As Long
'Integers...
Dim iWshsTHISWbk As Integer
'Bytes...
Dim yNmbrDimens As Byte
Dim yFileDimens As Byte
Dim yWshTypeDimens As Byte
'Next two used only for our status bar msg...
Dim yDataWshs As Byte
Dim yDataWshCounter As Byte
Dim yRowDimens As Byte
Dim yColDimens As Byte
Dim yCARTTtlCols As Byte
Dim i As Byte
```

```
FileHandler - 21

Dim j As Byte
Dim k As Byte
Dim m As Byte
Dim n As Byte
'Booleans...
Static bProcedurePreviouslyRun As Boolean
Dim bAddedToSkipRowColDict As Boolean
Dim bBADDataWsCreated As Boolean
Dim bWshsProtected As Boolean

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
'Check on protected wshs...
TrapForAndPromptReProtectedWshs _
        WshsProtectedRESULT:=bWshsProtected, _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
If bWshsProtected Then GoTo ExitPoint

'Double-check with user...
Dim ansImportData As VbMsgBoxResult
g_cMsgBoxHandler.AskImportDataQuest
ansImportData = g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbYesNoCancel)
If ansImportData = vbNo Or ansImportData = vbCancel Then GoTo ExitPoint

If Not g_bDebugMode Then Application.ScreenUpdating = False

'In case the user has made any changes to his source workbook, as a result of _
one of our prompts - or otherwise -...
If bProcedurePreviouslyRun Then g_wbSrcData.Save

yCARTTtlCols = m_yCART_META_COLS + m_yDIMENS_NON_AMT + m_yDIMENS_AMT
'Define some wshs & ranges...
Set wsValLists = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
Set rngValLstDataWshs = wsValLists.Range("vallstdodynWshsToImport")
Set rngValLstWshTypes = wsValLists.Range("vallstdodynWshTypes")
Set rngValTblWshNmTypeMap = wsValLists.Range("valtbldodynWshNmTypeMap")
Set wsDimenLocs = g_wbSrcData.Worksheets(g_sWSH_DIMEN_LOCS)
Set rngLocListDimNms = _
        wsDimenLocs.Range("tblDimenLocs" & Chr(91) & _
        g_sDIM_LOC_TBL_HDR_DIM_NM & Chr(93))
Set rngLocListSLRCols = _
        wsDimenLocs.Range("tblDimenLocs" & Chr(91) & _
        g_sDIM_LOC_TBL_HDR_SLR_COL & Chr(93))
Set rngLocListFileHC = _
        wsDimenLocs.Range("tblDimenLocs" & Chr(91) & _
        g_sDIM_LOC_TBL_HDR_FILE_HC & Chr(93))
Set rngLocListFileRR = _
        wsDimenLocs.Range("tblDimenLocs" & Chr(91) & _
        g_sDIM_LOC_TBL_HDR_FILE_RR & Chr(93))

yNmbrDimens = rngLocListDimNms.Rows.Count
yDataWshs = rngValLstDataWshs.Rows.Count
lCARTRowNext = 2

If g_bStepThruMode Then Stop
'Get hard-coded file dimensions...
For i = 1 To yNmbrDimens
    If StrComp(Trim(rngLocListFileHC.Cells(i, 1)), vbNullString) <> 0 Then
        If rngLocListSLRCols(i, 1) = 0 Then
            'We are working with one of our pseudo-dimensions, _
            Scale or Amt Type...
            If StrComp(Trim(rngLocListDimNms.Cells(i, 1)), _
                    g_sDIMEN_SCALE, vbTextCompare) = 0 Then
                Select Case Trim(rngLocListFileHC.Cells(i, 1))
                    Case g_sSCALE_THOUS
                        lScale = 1000#
                    Case g_sSCALE_MILLS
```

```
                    lScale = 1000000#
                Case g_sSCALE_BILLS
                    lScale = 1000000000#
                Case Else
                    lScale = 1
            End Select
        End If
    Else '...normal file dimensions
        'n = 0 on very first loop...
        ReDim Preserve tFileDimenVals(0 To n)
        tFileDimenVals(n).nm = rngLocListDimNms.Cells(i, 1)
        tFileDimenVals(n).SLRCol = _
                rngLocListDimNms.Offset(0, _
                g_yDT_COL_OFF_NM_2_SLRCOL).Cells(i, 1)
        tFileDimenVals(n).Val = rngLocListFileHC.Cells(i, 1)
        n = n + 1
    End If
End If
Next i
'Get range-ref defined file dimensions (DON'T RESET n!!!!!)...
Dim sFullAddr As String
Dim sWshNm As String
Dim sCellAddr As String
Dim yExclmtnPtPos As Byte
If g_bStepThruMode Then Stop
For i = 1 To yNmbrDimens
    If StrComp(Trim(rngLocListFileRR.Cells(i, 1)), vbNullString) <> 0 Then
        ReDim Preserve tFileDimenVals(0 To n)
        'If g_bStepThruMode Then Stop
        sFullAddr = Trim(rngLocListFileRR.Cells(i, 1))
        yExclmtnPtPos = InStr(1, sFullAddr, Chr(33))
        sWshNm = Left(sFullAddr, yExclmtnPtPos - 1)
        'Trim apostrophes, which occur if a worksheet name has embedded spaces...
        sWshNm = Replace(sWshNm, Chr(39), "")
        sCellAddr = Right(sFullAddr, Len(sFullAddr) - yExclmtnPtPos)
        'Trap for an empty cell...
        If Len(Trim(g_wbSrcData.Worksheets(sWshNm).Range(sCellAddr))) = 0 Then
            g_wbSrcData.Activate
            Worksheets(sWshNm).Activate
            Worksheets(sWshNm).Range(sCellAddr).Activate
            Application.ScreenUpdating = True
            g_cMsgBoxHandler.SetEmptyFileDimenCellMsg _
                    DimenNm:=rngLocListDimNms.Cells(i, 1), _
                    FileDimenCellAddr:=sFullAddr
            g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbInformation
            GoTo ExitPoint
        End If
        tFileDimenVals(n).nm = rngLocListDimNms.Cells(i, 1)
        tFileDimenVals(n).SLRCol = _
                rngLocListDimNms.Offset(0, g_yDT_COL_OFF_NM_2_SLRCOL).Cells(i, 1)
        tFileDimenVals(n).Val = _
                g_wbSrcData.Worksheets(sWshNm).Range(sCellAddr)
        n = n + 1
    End If
Next i
yFileDimens = n '...it was incremented by 1 at the end of the loop

'*****LOOP THROUGH WSH TYPES...
If g_bStepThruMode Then Stop
For i = 1 To rngValLstWshTypes.Rows.Count
    '***Variable (re-)declarations...
    'Custom types...
    'We're cheating a bit by using our custom type here, but it works...
    Dim tWshTypeDimNmAndAddr() As m_TypDimNmVal
    Dim tColDimenRows() As m_TypDimNmRC
    Dim tRowDimenCols() As m_TypDimNmRC
    'Strings...
    Dim sWshType As String

    'Reset some variables...
    yRowDimens = 0
```

```
    yColDimens = 0
    lDataRowFirst = 0
    lDataColFirst = 0
    yWshTypeDimens = 0

    sWshType = Trim(rngValLstWshTypes.Cells(i, 1))
    Set rngLocListCurrWshType = _
            wsDimenLocs.Range("tblDimenLocs" & Chr(91) & sWshType & Chr(93))
    '**** POPULATE THE 3 SETS OF DIMENSIONS FOR OUR WSH TYPE...
    If g_bStepThruMode Then Stop
    For j = 1 To rngLocListCurrWshType.Rows.Count
        sCellAddr = Trim(rngLocListCurrWshType.Cells(j, 1))
        If StrComp(sCellAddr, vbNullString) <> 0 Then
            'Trap for those entries (at the moment only Scale) that are in our _
            table but are not really dimensions...
            If rngLocListSLRCols(j, 1) = 0 Then
                'These are our psuedo-dimensions, so...
                If StrComp(Trim(rngLocListDimNms.Cells(j, 1)), _
                        g_sDIMEN_SCALE, vbTextCompare) = 0 Then
                    'This will NOT be a cell addr, but we'll use the variable...
                    Select Case sCellAddr
                        Case g_sSCALE_THOUS
                            lScale = 1000#
                        Case g_sSCALE_MILLS
                            lScale = 1000000#
                        Case g_sSCALE_BILLS
                            lScale = 1000000000#
                        Case Else
                            lScale = 1
                    End Select
                End If
            Else
                If Range(sCellAddr).Rows.Count > 1 Then
                    'Refers to a column (i.e., row dimension)...
                    ReDim Preserve tRowDimenCols(0 To yRowDimens)
                    tRowDimenCols(yRowDimens).nm = _
                            rngLocListDimNms.Cells(j, 1)
                    tRowDimenCols(yRowDimens).SLRCol = _
                            rngLocListSLRCols.Cells(j, 1)
                    tRowDimenCols(yRowDimens).RowCol = _
                            Range(sCellAddr).Column
                    'Track first data column...
                    If Range(sCellAddr).Column + 1 > lDataColFirst Then _
                            lDataColFirst = Range(sCellAddr).Column + 1
                    yRowDimens = yRowDimens + 1
                ElseIf Range(sCellAddr).Columns.Count > 1 Then
                    'Refers to a row (i.e., col dimension)...
                    ReDim Preserve tColDimenRows(0 To yColDimens)
                    tColDimenRows(yColDimens).nm = _
                            rngLocListDimNms.Cells(j, 1)
                    tColDimenRows(yColDimens).SLRCol = _
                            rngLocListSLRCols.Cells(j, 1)
                    tColDimenRows(yColDimens).RowCol = _
                            Range(sCellAddr).Row
                    'Track first data row...
                    'If g_bDebugMode Then Stop
                    If Range(sCellAddr).Row + 1 > lDataRowFirst Then _
                            lDataRowFirst = Range(sCellAddr).Row + 1
                    yColDimens = yColDimens + 1
                Else '...we have a single-cell address (i.e., wsh dimension)
                    ReDim Preserve tWshTypeDimNmAndAddr(0 To yWshTypeDimens)
                    tWshTypeDimNmAndAddr(yWshTypeDimens).nm = _
                            rngLocListDimNms.Cells(j, 1)
                    tWshTypeDimNmAndAddr(yWshTypeDimens).SLRCol = _
                            rngLocListDimNms. _
                            Offset(0, g_yDT_COL_OFF_NM_2_SLRCOL).Cells(j, 1)
                    tWshTypeDimNmAndAddr(yWshTypeDimens).Val = _
                            rngLocListCurrWshType.Cells(j, 1)
                    yWshTypeDimens = yWshTypeDimens + 1
                End If
            End If
```

```
        End If
    Next j
    '*****LOOP THROUGH WORKSHEETS...
    If g_bStepThruMode Then Stop
    For j = 1 To rngValTblWshNmTypeMap.Rows.Count
        '***Variable (re-)declarations...
        'Custom types...
        Dim tWshTypeDimenVals() As m_TypDimNmVal
        'Variants...
        Dim vaDataSrc() As Variant
        Dim vaFunction() As Variant
        Dim vaLedger() As Variant
        Dim vaLglEnt() As Variant
        Dim vaMgdBus() As Variant
        Dim vaMVAcct() As Variant
        Dim vaOpLines() As Variant
        Dim vaOrigGeog() As Variant
        Dim vaPeriod() As Variant
        Dim vaProduct() As Variant
        Dim vaScenario() As Variant
        Dim vaYear() As Variant
        Dim vaAmts() As Variant
        'Ranges...
        Dim rngWshUsedRng As Range
        Dim rngLastCell As Range
        Dim rngRowDimenLbls As Range
        Dim rngColDimenHdrs As Range
        Dim rngSrcWsData As Range
        Dim rngDimenCell As Range
        Dim c As Range
        'Strings...
        Dim sCurrDataWsh As String
        Dim sDimen As String
        Dim sSBMsgWsh As String
        Dim sColDimen As String
        Dim sRowDimen As String

        If StrComp(sWshType, _
                (rngValTblWshNmTypeMap.Cells(j, g_yWSH_ASSOC_TBL_TYPE_COL)), _
                vbTextCompare) <> 0 Then GoTo DoneWithWsh
        'Set the data worksheet...
        sCurrDataWsh = _
                rngValTblWshNmTypeMap.Cells(j, g_yWSH_ASSOC_TBL_WSH_NM_COL)
        Set wsSrcData = g_wbSrcData.Worksheets(sCurrDataWsh)
        yDataWshCounter = yDataWshCounter + 1
        sSBMsgWsh = _
                "Processing worksheet: " & sCurrDataWsh & _
                " (" & yDataWshCounter & " of " & yDataWshs & ")"
        Application.StatusBar = sSBMsgWsh
        'Reset some stuff...
        Set wsBADData = Nothing
        bBADDataWsCreated = False
        Set dictBadHiddenOrEmptyRows = New Dictionary
        Set dictBadHiddenOrEmptyCols = New Dictionary
        lDataColLast = 0
        lDataRowLast = 0
        'Populate our wsh dimens (if they exist!)...
        If yWshTypeDimens > 0 Then
            ReDim tWshTypeDimenVals(0 To yWshTypeDimens - 1)
            For k = 1 To yWshTypeDimens
                'Trap for an empty cell...
                If Len(Trim(wsSrcData.Range(tWshTypeDimNmAndAddr(k - 1).Val))) = 0 Then
                    'Prompt the user...
                    g_wbSrcData.Activate
                    Worksheets(sCurrDataWsh).Activate
                    Worksheets(sCurrDataWsh).Range(tWshTypeDimNmAndAddr(k - 1).Val).Activate
                    Application.ScreenUpdating = True
                    g_cMsgBoxHandler.SetEmptyWshDimenCellMsg _
                            WshTypeNm:=sWshType, _
                            WshNm:=sCurrDataWsh, _
                            DimenNm:=tWshTypeDimNmAndAddr(k - 1).nm, _
```

```
                        WshDimenCellAddr:=tWshTypeDimNmAndAddr(k - 1).Val
                g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbInformation
                If lCARTRowNext > 2 Then
                    RemoveImportedDataAndDeleteBADWshs _
                            LastDataRow:=lCARTRowNext - 1, _
                            RanOkRESULT:=bCalledProcedureRanOk
                        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
                End If
                GoTo ExitPoint
            End If
            tWshTypeDimenVals(k - 1).nm = tWshTypeDimNmAndAddr(k - 1).nm
            tWshTypeDimenVals(k - 1).SLRCol = _
                    tWshTypeDimNmAndAddr(k - 1).SLRCol
            tWshTypeDimenVals(k - 1).Val = _
                    wsSrcData.Range(tWshTypeDimNmAndAddr(k - 1).Val)
        Next k
    End If
    'Find the last cell in the data worksheet...
    If g_bStepThruMode Then Stop
    wsSrcData.Activate
    'NOTE:  Interestingly, this line (original code) fails when used on _
    a protected wsh...
    'Set rngLastCell = wsSrcData.UsedRange.SpecialCells(xlCellTypeLastCell)
    'And this line works...
    Set rngWshUsedRng = wsSrcData.Range(wsSrcData.UsedRange.Address)
    'But now, of course, we have to resize it...
    Set rngLastCell = rngWshUsedRng.Resize(1, 1)
    Set rngLastCell = rngLastCell.Offset(rngWshUsedRng.Rows.Count - 1, _
            rngWshUsedRng.Columns.Count - 1)
    lDataColLastPrelim = rngLastCell.Column
    lDataRowLastPrelim = rngLastCell.Row
    'As always, we now need to see if this really IS the last cell...
    If Len(Trim(rngLastCell)) = 0 Then
        'Find the RH most column in our column dimension rows...
        For k = 1 To yColDimens
            wsSrcData.Cells(tColDimenRows(k - 1).RowCol, _
                    lDataColLastPrelim).Activate
            If Len(Trim(ActiveCell)) > 0 Then
                lDataColLast = lDataColLastPrelim
                k = yColDimens '...break loop
            Else
                ActiveCell.End(xlToLeft).Activate
                If ActiveCell.Column > lDataColLast Then _
                        lDataColLast = ActiveCell.Column
            End If
        Next k
        'Find the bottom row in our row dimension columns...
        For k = 1 To yRowDimens
            wsSrcData.Cells(lDataRowLastPrelim, _
                    tRowDimenCols(k - 1).RowCol).Activate
            If Len(Trim(ActiveCell)) > 0 Then
                lDataRowLast = lDataRowLastPrelim
                k = yRowDimens '...break loop
            Else
                ActiveCell.End(xlUp).Activate
                If ActiveCell.Row > lDataRowLast Then _
                        lDataRowLast = ActiveCell.Row
            End If
        Next k
    Else
        lDataRowLast = lDataRowLastPrelim
        lDataColLast = lDataColLastPrelim
    End If
    'Set up our array for data going into CART sheet...
    '****************
    'POPULATE OUR DATA ARRAYS...
    '****************
    'Row dimension arrays...
    For k = 0 To yRowDimens - 1
        'Reset...
        lRowDimenCol = tRowDimenCols(k).RowCol
```

```
            sDimen = Trim(tRowDimenCols(k).nm)
            Set rngRowDimenLbls = _
                    wsSrcData.Range(Cells(lDataRowFirst, lRowDimenCol), _
                            Cells(lDataRowLast, lRowDimenCol))
           'In the following case statements we must trap for single row _
            data files, because assigning a range.Value2 to a 2-dimensional _
            variant array will fail when the range is only a single cell _
            (and all RowDimenLbls are, of course, already just a single col)...
            'If g_bStepThruMode Then Stop
            Select Case sDimen
                Case g_sDIMEN_DATA_SRC
                    If rngRowDimenLbls.Rows.Count = 1 Then
                        ReDim vaDataSrc(1 To 1, 1 To 1) As Variant
                        vaDataSrc(1, 1) = rngRowDimenLbls.Cells(1, 1)
                    Else
                        vaDataSrc = rngRowDimenLbls.Value2
                    End If
                Case g_sDIMEN_FUNC
                    If rngRowDimenLbls.Rows.Count = 1 Then
                        ReDim vaFunction(1 To 1, 1 To 1) As Variant
                        vaFunction(1, 1) = rngRowDimenLbls.Cells(1, 1)
                    Else
                        vaFunction = rngRowDimenLbls.Value2
                    End If
                Case g_sDIMEN_LDGR
                    If rngRowDimenLbls.Rows.Count = 1 Then
                        ReDim vaLedger(1 To 1, 1 To 1) As Variant
                        vaLedger(1, 1) = rngRowDimenLbls.Cells(1, 1)
                    Else
                        vaLedger = rngRowDimenLbls.Value2
                    End If
                Case g_sDIMEN_LGL_ENT
                    If rngRowDimenLbls.Rows.Count = 1 Then
                        ReDim vaLglEnt(1 To 1, 1 To 1) As Variant
                        vaLglEnt(1, 1) = rngRowDimenLbls.Cells(1, 1)
                    Else
                        vaLglEnt = rngRowDimenLbls.Value2
                    End If
                Case g_sDIMEN_MGD_BUS
                    If rngRowDimenLbls.Rows.Count = 1 Then
                        ReDim vaMgdBus(1 To 1, 1 To 1) As Variant
                        vaMgdBus(1, 1) = rngRowDimenLbls.Cells(1, 1)
                    Else
                        vaMgdBus = rngRowDimenLbls.Value2
                    End If
                Case g_sDIMEN_MV_ACCT
                    If rngRowDimenLbls.Rows.Count = 1 Then
                        ReDim vaMVAcct(1 To 1, 1 To 1) As Variant
                        vaMVAcct(1, 1) = rngRowDimenLbls.Cells(1, 1)
                    Else
                        vaMVAcct = rngRowDimenLbls.Value2
                    End If
                Case g_sDIMEN_OPER_LNS
                    If rngRowDimenLbls.Rows.Count = 1 Then
                        ReDim vaOpLines(1 To 1, 1 To 1) As Variant
                        vaOpLines(1, 1) = rngRowDimenLbls.Cells(1, 1)
                    Else
                        vaOpLines = rngRowDimenLbls.Value2
                    End If
                Case g_sDIMEN_ORIG_GEOG
                    If rngRowDimenLbls.Rows.Count = 1 Then
                        ReDim vaOrigGeog(1 To 1, 1 To 1) As Variant
                        vaOrigGeog(1, 1) = rngRowDimenLbls.Cells(1, 1)
                    Else
                        vaOrigGeog = rngRowDimenLbls.Value2
                    End If
                Case g_sDIMEN_PER
                    If rngRowDimenLbls.Rows.Count = 1 Then
                        ReDim vaPeriod(1 To 1, 1 To 1) As Variant
                        vaPeriod(1, 1) = rngRowDimenLbls.Cells(1, 1)
                    Else
```

```
                    vaPeriod = rngRowDimenLbls.Value2
                End If
            Case g_sDIMEN_PROD
                If rngRowDimenLbls.Rows.Count = 1 Then
                    ReDim vaProduct(1 To 1, 1 To 1) As Variant
                    vaProduct(1, 1) = rngRowDimenLbls.Cells(1, 1)
                Else
                    vaProduct = rngRowDimenLbls.Value2
                End If
            Case g_sDIMEN_SCENAR
                If rngRowDimenLbls.Rows.Count = 1 Then
                    ReDim vaScenario(1 To 1, 1 To 1) As Variant
                    vaScenario(1, 1) = rngRowDimenLbls.Cells(1, 1)
                Else
                    vaScenario = rngRowDimenLbls.Value2
                End If
            Case g_sDIMEN_YR
                If rngRowDimenLbls.Rows.Count = 1 Then
                    ReDim vaYear(1 To 1, 1 To 1) As Variant
                    vaYear(1, 1) = rngRowDimenLbls.Cells(1, 1)
                Else
                    vaYear = rngRowDimenLbls.Value2
                End If
        End Select
    Next k
    'Column dimension arrays...
    For k = 0 To yColDimens - 1
        lColDimenRow = tColDimenRows(k).RowCol
        sDimen = Trim(tColDimenRows(k).nm)
        Set rngColDimenHdrs = _
                wsSrcData.Range(Cells(lColDimenRow, lDataColFirst), _
                        Cells(lColDimenRow, lDataColLast))
        'In the following case statements we must trap for single column _
        data files, because assigning a range.Value2 to a 2-dimensional _
        variant array will fail when the range is only a single cell _
        (and all ColDimenRows are, of course, already just a single row)...
        Select Case sDimen
            Case g_sDIMEN_DATA_SRC
                If rngColDimenHdrs.Columns.Count = 1 Then
                    ReDim vaDataSrc(1 To 1, 1 To 1) As Variant
                    vaDataSrc(1, 1) = rngColDimenHdrs.Cells(1, 1)
                Else
                    vaDataSrc = rngColDimenHdrs.Value2
                End If
            Case g_sDIMEN_FUNC
                If rngColDimenHdrs.Columns.Count = 1 Then
                    ReDim vaFunction(1 To 1, 1 To 1) As Variant
                    vaFunction(1, 1) = rngColDimenHdrs.Cells(1, 1)
                Else
                    vaFunction = rngColDimenHdrs.Value2
                End If
            Case g_sDIMEN_LDGR
                If rngColDimenHdrs.Columns.Count = 1 Then
                    ReDim vaLedger(1 To 1, 1 To 1) As Variant
                    vaLedger(1, 1) = rngColDimenHdrs.Cells(1, 1)
                Else
                    vaLedger = rngColDimenHdrs.Value2
                End If
            Case g_sDIMEN_LGL_ENT
                If rngColDimenHdrs.Columns.Count = 1 Then
                    ReDim vaLglEnt(1 To 1, 1 To 1) As Variant
                    vaLglEnt(1, 1) = rngColDimenHdrs.Cells(1, 1)
                Else
                    vaLglEnt = rngColDimenHdrs.Value2
                End If
            Case g_sDIMEN_MGD_BUS
                If rngColDimenHdrs.Columns.Count = 1 Then
                    ReDim vaMgdBus(1 To 1, 1 To 1) As Variant
                    vaMgdBus(1, 1) = rngColDimenHdrs.Cells(1, 1)
                Else
                    vaMgdBus = rngColDimenHdrs.Value2
```

```
                End If
            Case g_sDIMEN_MV_ACCT
                If rngColDimenHdrs.Columns.Count = 1 Then
                    ReDim vaMVAcct(1 To 1, 1 To 1) As Variant
                    vaMVAcct(1, 1) = rngColDimenHdrs.Cells(1, 1)
                Else
                    vaMVAcct = rngColDimenHdrs.Value2
                End If
            Case g_sDIMEN_OPER_LNS
                If rngColDimenHdrs.Columns.Count = 1 Then
                    ReDim vaOpLines(1 To 1, 1 To 1) As Variant
                    vaOpLines(1, 1) = rngColDimenHdrs.Cells(1, 1)
                Else
                    vaOpLines = rngColDimenHdrs.Value2
                End If
            Case g_sDIMEN_ORIG_GEOG
                If rngColDimenHdrs.Columns.Count = 1 Then
                    ReDim vaOrigGeog(1 To 1, 1 To 1) As Variant
                    vaOrigGeog(1, 1) = rngColDimenHdrs.Cells(1, 1)
                Else
                    vaOrigGeog = rngColDimenHdrs.Value2
                End If
            Case g_sDIMEN_PER
                If rngColDimenHdrs.Columns.Count = 1 Then
                    ReDim vaPeriod(1 To 1, 1 To 1) As Variant
                    vaPeriod(1, 1) = rngColDimenHdrs.Cells(1, 1)
                Else
                    vaPeriod = rngColDimenHdrs.Value2
                End If
            Case g_sDIMEN_PROD
                If rngColDimenHdrs.Columns.Count = 1 Then
                    ReDim vaProduct(1 To 1, 1 To 1) As Variant
                    vaProduct(1, 1) = rngColDimenHdrs.Cells(1, 1)
                Else
                    vaProduct = rngColDimenHdrs.Value2
                End If
            Case g_sDIMEN_SCENAR
                If rngColDimenHdrs.Columns.Count = 1 Then
                    ReDim vaScenario(1 To 1, 1 To 1) As Variant
                    vaScenario(1, 1) = rngColDimenHdrs.Cells(1, 1)
                Else
                    vaScenario = rngColDimenHdrs.Value2
                End If
            Case g_sDIMEN_YR
                If rngColDimenHdrs.Columns.Count = 1 Then
                    ReDim vaYear(1 To 1, 1 To 1) As Variant
                    vaYear(1, 1) = rngColDimenHdrs.Cells(1, 1)
                Else
                    vaYear = rngColDimenHdrs.Value2
                End If
        End Select
    Next k
    'Identify our bad/empty/hidden rows...
    If g_bStepThruMode Then Stop
    For lRowNo = lDataRowFirst To lDataRowLast
        bAddedToSkipRowColDict = False '...reset
        'Test for whether row is hidden...
        If wsSrcData.Rows(lRowNo).EntireRow.Hidden = True Then
            dictBadHiddenOrEmptyRows.Add lRowNo, lRowNo
            bAddedToSkipRowColDict = True
        End If
        'Test for whether row is empty...
        If IsEmpty(wsSrcData.Range(Cells(lRowNo, lDataColFirst), _
                Cells(lRowNo, lDataColLast))) And _
                Not bAddedToSkipRowColDict Then
            dictBadHiddenOrEmptyRows.Add lRowNo, lRowNo
            bAddedToSkipRowColDict = True
        End If
        'If g_bStepThruMode Then Stop
        'Test for missing mandatory dimension or an error code in _
        the row labels...
```

```
            If Not bAddedToSkipRowColDict Then
                For k = 0 To yRowDimens - 1
                    'Reset...
                    lRowDimenCol = 0
                    lRowDimenCol = tRowDimenCols(k).RowCol
                    'If the cell is part of a merged range get the value _
                    being displayed in the merged range...
                    Set rngDimenCell = wsSrcData.Cells(lRowNo, lRowDimenCol)
                    If rngDimenCell.MergeArea.Rows.Count > 1 Then
                        sRowDimen = GetValueInMergedCells( _
                                MergedCell:=rngDimenCell, _
                                Scope:=projScopeRow, _
                                RanOkRESULT:=bCalledProcedureRanOk)
                            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
                        'Now handle a case of merged cells with no values inside...
                        If StrComp(sRowDimen, vbNullString) = 0 Then
                            dictBadHiddenOrEmptyRows.Add lRowNo, lRowNo
                            bAddedToSkipRowColDict = True
                            k = yRowDimens - 1 '...break loop
                        Else
                            'Push the row dimension into the appropriate array...
                            Select Case Trim(tRowDimenCols(k).nm)
                                Case g_sDIMEN_DATA_SRC
                                    vaDataSrc(1 + lRowNo - lDataRowFirst, 1) = _
                                            sRowDimen
                                Case g_sDIMEN_FUNC
                                    vaFunction(1 + lRowNo - lDataRowFirst, 1) = _
                                            sRowDimen
                                Case g_sDIMEN_LDGR
                                    vaLedger(1 + lRowNo - lDataRowFirst, 1) = _
                                            sRowDimen
                                Case g_sDIMEN_LGL_ENT
                                    vaLglEnt(1 + lRowNo - lDataRowFirst, 1) = _
                                            sRowDimen
                                Case g_sDIMEN_MGD_BUS
                                    vaMgdBus(1 + lRowNo - lDataRowFirst, 1) = _
                                            sRowDimen
                                Case g_sDIMEN_MV_ACCT
                                    vaMVAcct(1 + lRowNo - lDataRowFirst, 1) = _
                                            sRowDimen
                                Case g_sDIMEN_OPER_LNS
                                    vaOpLines(1 + lRowNo - lDataRowFirst, 1) = _
                                            sRowDimen
                                Case g_sDIMEN_ORIG_GEOG
                                    vaOrigGeog(1 + lRowNo - lDataRowFirst, 1) = _
                                            sRowDimen
                                Case g_sDIMEN_PER
                                    vaPeriod(1 + lRowNo - lDataRowFirst, 1) = _
                                            sRowDimen
                                Case g_sDIMEN_PROD
                                    vaProduct(1 + lRowNo - lDataRowFirst, 1) = _
                                            sRowDimen
                                Case g_sDIMEN_SCENAR
                                    vaScenario(1 + lRowNo - lDataRowFirst, 1) = _
                                            sRowDimen
                                Case g_sDIMEN_YR
                                    vaYear(1 + lRowNo - lDataRowFirst, 1) = _
                                            sRowDimen
                            End Select
                        End If
                    Else
                        'Trap for error or empty cells...
                        If IsError(wsSrcData.Cells(lRowNo, lRowDimenCol)) Or _
                                Len(Trim(wsSrcData.Cells( _
                                lRowNo, lRowDimenCol))) = 0 Then
                            dictBadHiddenOrEmptyRows.Add lRowNo, lRowNo
                            bAddedToSkipRowColDict = True
                            k = yRowDimens - 1 '...break loop
                        End If
                    End If
                Next k
```

```
            End If
        Next lRowNo
        If g_bStepThruMode Then Stop
       'Identify our bad/empty/hidden columns...
        For lColNo = lDataColFirst To lDataColLast
            bAddedToSkipRowColDict = False '...reset
            'Test for whether col is hidden...
            If wsSrcData.Columns(lColNo).EntireColumn.Hidden = True Then
                dictBadHiddenOrEmptyCols.Add lColNo, lColNo
                bAddedToSkipRowColDict = True
            End If
            'Test for whether col is empty...
            If IsEmpty(wsSrcData.Range(Cells(lDataRowFirst, lColNo), _
                    Cells(lDataRowLast, lColNo))) And _
                    Not bAddedToSkipRowColDict Then
                dictBadHiddenOrEmptyCols.Add lColNo, lColNo
                bAddedToSkipRowColDict = True
            End If
            'If g_bStepThruMode Then Stop
            'Test for missing mandatory dimension...
            If Not bAddedToSkipRowColDict Then
                For k = 0 To yColDimens - 1
                    'Reset...
                    lColDimenRow = 0
                    lColDimenRow = tColDimenRows(k).RowCol
                    'If the cell is part of a merged range get the value _
                    being displayed in the merged range...
                    Set rngDimenCell = wsSrcData.Cells(lColDimenRow, lColNo)
                    If rngDimenCell.MergeArea.Columns.Count > 1 Then
                        sColDimen = GetValueInMergedCells( _
                                MergedCell:=rngDimenCell, _
                                Scope:=projScopeCol, _
                                RanOkRESULT:=bCalledProcedureRanOk)
                            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
                        'Now handle a case of merged cells with no values inside...
                        If StrComp(sColDimen, vbNullString) = 0 Then
                            dictBadHiddenOrEmptyCols.Add lColNo, lColNo
                            bAddedToSkipRowColDict = True
                            k = yColDimens - 1 '...break loop
                        Else
                            Select Case Trim(tColDimenRows(k).nm)
                                Case g_sDIMEN_DATA_SRC
                                    vaDataSrc(1, 1 + lColNo - lDataColFirst) = _
                                            sColDimen
                                Case g_sDIMEN_FUNC
                                    vaFunction(1, 1 + lColNo - lDataColFirst) = _
                                            sColDimen
                                Case g_sDIMEN_LDGR
                                    vaLedger(1, 1 + lColNo - lDataColFirst) = _
                                            sColDimen
                                Case g_sDIMEN_LGL_ENT
                                    vaLglEnt(1, 1 + lColNo - lDataColFirst) = _
                                            sColDimen
                                Case g_sDIMEN_MGD_BUS
                                    vaMgdBus(1, 1 + lColNo - lDataColFirst) = _
                                            sColDimen
                                Case g_sDIMEN_MV_ACCT
                                    vaMVAcct(1, 1 + lColNo - lDataColFirst) = _
                                            sColDimen
                                Case g_sDIMEN_OPER_LNS
                                    vaOpLines(1, 1 + lColNo - lDataColFirst) = _
                                            sColDimen
                                Case g_sDIMEN_ORIG_GEOG
                                    vaOrigGeog(1, 1 + lColNo - lDataColFirst) = _
                                            sColDimen
                                Case g_sDIMEN_PER
                                    vaPeriod(1, 1 + lColNo - lDataColFirst) = _
                                            sColDimen
                                Case g_sDIMEN_PROD
                                    vaProduct(1, 1 + lColNo - lDataColFirst) = _
                                            sColDimen
```

```
                            Case g_sDIMEN_SCENAR
                                vaScenario(1, 1 + lColNo - lDataColFirst) = _
                                        sColDimen
                            Case g_sDIMEN_YR
                                vaYear(1, 1 + lColNo - lDataColFirst) = _
                                        sColDimen
                        End Select
                    End If
                Else
                    If IsError(wsSrcData.Cells(lColDimenRow, lColNo)) Or _
                            Len(Trim(wsSrcData.Cells( _
                            lColDimenRow, lColNo))) = 0 Then
                        dictBadHiddenOrEmptyCols.Add lColNo, lColNo
                        bAddedToSkipRowColDict = True
                        k = yColDimens - 1 '...break loop
                    End If
                End If
            Next k
        End If
    Next lColNo
    If g_bStepThruMode Then Stop
    'Populate our Amounts array...
    Set rngSrcWsData = wsSrcData.Range(Cells(lDataRowFirst, lDataColFirst), _
            Cells(lDataRowLast, lDataColLast))
    vaAmts = rngSrcWsData.Value2
    '*****************************************
    'Now populate our array that will wind CART data file...
    '*****************************************
    lDataRows = 1 + lDataRowLast - lDataRowFirst
    lDataCols = 1 + lDataColLast - lDataColFirst
    lCARTWshRows = lDataRows * lDataCols
    Dim vaCARTWshData() As Variant
    ReDim vaCARTWshData(1 To lCARTWshRows, 1 To yCARTTtlCols)
    lCARTDataRow = 1
    '***** LOOP THROUGH DATA ROWS...
    For x = lDataRowFirst To lDataRowLast
        'First see whether we are even going to be creating a CART row...
        If dictBadHiddenOrEmptyRows.Exists(x) Then
            'Take care of flagging things in the BAD data wsh...
            If Not bBADDataWsCreated Then
                iWshsTHISWbk = ThisWorkbook.Sheets.Count
                'In case we wind up with duplicate range names...
                Application.DisplayAlerts = False
                wsSrcData.Copy _
                        after:=ThisWorkbook.Sheets(iWshsTHISWbk)
                Application.DisplayAlerts = True
                'Rename wsh...
                Set wsBADData = _
                        ThisWorkbook.Worksheets(sCurrDataWsh)
                wsBADData.name = _
                    Left(sCurrDataWsh, _
                    yXL_MAX_WSH_NM - Len(m_sBAD_WS_SUFFIX) - _
                    yDIGITS_IN_BAD_WSH_NM_COUNTER) & _
                    m_sBAD_WS_SUFFIX & Format(yDataWshCounter, "00")
                bBADDataWsCreated = True
            End If
            For lRowDimenCol = 1 To yRowDimens
                y = tRowDimenCols(lRowDimenCol - 1).RowCol
                With wsBADData.Cells(x, y)
                    .Interior.ColorIndex = 1
                    .Font.ColorIndex = 2
                End With
            Next lRowDimenCol
            GoTo DoneWithRow
        End If
        'We are processing row, so populate row dimensions in array...
        Dim tRowDimenVals() As m_TypDimNmVal
        ReDim tRowDimenVals(0 To yRowDimens - 1)
        'Populate our row dimensions...
        For m = 0 To yRowDimens - 1
            lRowDimenCol = tRowDimenCols(m).RowCol
```

```
        sDimen = Trim(tRowDimenCols(m).nm)
        Select Case sDimen
        Case g_sDIMEN_DATA_SRC
            tRowDimenVals(m).SLRCol = g_yCOL_DIMEN_DATA_SRC
            tRowDimenVals(m).Val = vaDataSrc(1 + x - lDataRowFirst, 1)
        Case g_sDIMEN_FUNC
            tRowDimenVals(m).SLRCol = g_yCOL_DIMEN_FUNC
            tRowDimenVals(m).Val = vaFunction(1 + x - lDataRowFirst, 1)
        Case g_sDIMEN_LDGR
            tRowDimenVals(m).SLRCol = g_yCOL_DIMEN_LDGR
            tRowDimenVals(m).Val = vaLedger(1 + x - lDataRowFirst, 1)
        Case g_sDIMEN_LGL_ENT
            tRowDimenVals(m).SLRCol = g_yCOL_DIMEN_LGL_ENT
            tRowDimenVals(m).Val = vaLglEnt(1 + x - lDataRowFirst, 1)
        Case g_sDIMEN_MGD_BUS
            tRowDimenVals(m).SLRCol = g_yCOL_DIMEN_MGD_BUS
            tRowDimenVals(m).Val = vaMgdBus(1 + x - lDataRowFirst, 1)
        Case g_sDIMEN_MV_ACCT
            tRowDimenVals(m).SLRCol = g_yCOL_DIMEN_MV_ACCT
            tRowDimenVals(m).Val = vaMVAcct(1 + x - lDataRowFirst, 1)
        Case g_sDIMEN_OPER_LNS
            tRowDimenVals(m).SLRCol = g_yCOL_DIMEN_OPER_LNS
            tRowDimenVals(m).Val = vaOpLines(1 + x - lDataRowFirst, 1)
        Case g_sDIMEN_ORIG_GEOG
            tRowDimenVals(m).SLRCol = g_yCOL_DIMEN_ORIG_GEOG
            tRowDimenVals(m).Val = vaOrigGeog(1 + x - lDataRowFirst, 1)
        Case g_sDIMEN_PER
            tRowDimenVals(m).SLRCol = g_yCOL_DIMEN_PER
            tRowDimenVals(m).Val = vaPeriod(1 + x - lDataRowFirst, 1)
        Case g_sDIMEN_PROD
            tRowDimenVals(m).SLRCol = g_yCOL_DIMEN_PROD
            tRowDimenVals(m).Val = vaProduct(1 + x - lDataRowFirst, 1)
        Case g_sDIMEN_SCENAR
            tRowDimenVals(m).SLRCol = g_yCOL_DIMEN_SCENAR
            tRowDimenVals(m).Val = vaScenario(1 + x - lDataRowFirst, 1)
        Case g_sDIMEN_YR
            tRowDimenVals(m).SLRCol = g_yCOL_DIMEN_YR
            tRowDimenVals(m).Val = vaYear(1 + x - lDataRowFirst, 1)
        End Select
    Next m
    '***** LOOP THROUGH COLUMNS IN DATA ROW...
    For y = lDataColFirst To lDataColLast
        'Reset some variables...
        dRawAmt = 0
        cAdjAmt = 0
        lBadHdrRow = 0
        'See whether the data column is one we want or not...
        If dictBadHiddenOrEmptyCols.Exists(y) Then
            'Take care of flagging things in the BAD data wsh...
            If Not bBADDataWsCreated Then
                iWshsTHISWbk = ThisWorkbook.Sheets.Count
                'In case we wind up with duplicate range names...
                Application.DisplayAlerts = False
                wsSrcData.Copy _
                        after:=ThisWorkbook.Sheets(iWshsTHISWbk)
                Application.DisplayAlerts = True
                'Rename wsh...
                Set wsBADData = _
                        ThisWorkbook.Worksheets(sCurrDataWsh)
                wsBADData.name = _
                    Left(sCurrDataWsh, _
                    yXL_MAX_WSH_NM - Len(m_sBAD_WS_SUFFIX) - _
                    yDIGITS_IN_BAD_WSH_NM_COUNTER) & _
                    m_sBAD_WS_SUFFIX & Format(yDataWshCounter, "00")
                bBADDataWsCreated = True
            End If
            For lColDimenRow = 1 To yColDimens
                lBadHdrRow = tColDimenRows(lColDimenRow - 1).RowCol
                With wsBADData.Cells(lBadHdrRow, y)
                    .Interior.ColorIndex = 1
                    .Font.ColorIndex = 2
```

```
                End With
            Next lColDimenRow
            GoTo DoneWithCol
        End If
        'Test for no data...
        If IsEmpty(vaAmts(1 + x - lDataRowFirst, 1 + y - lDataColFirst)) _
            Then GoTo DoneWithCol
        'Test for BAD data...
        If Not IsNumeric(vaAmts(1 + x - lDataRowFirst, 1 + y - lDataColFirst)) Then
            'Remove the entry from our array...
            vaAmts(1 + x - lDataRowFirst, 1 + y - lDataColFirst) = vbNullString
            'Take care of flagging things in the BAD data wsh...
            If Not bBADDataWsCreated Then
                iWshsTHISWbk = ThisWorkbook.Sheets.Count
                'In case we wind up with duplicate range names...
                Application.DisplayAlerts = False
                wsSrcData.Copy _
                        after:=ThisWorkbook.Sheets(iWshsTHISWbk)
                Application.DisplayAlerts = True
                'Rename wsh...
                Set wsBADData = _
                        ThisWorkbook.Worksheets(sCurrDataWsh)
                wsBADData.name = _
                    Left(sCurrDataWsh, _
                    yXL_MAX_WSH_NM - Len(m_sBAD_WS_SUFFIX) - _
                    yDIGITS_IN_BAD_WSH_NM_COUNTER) & _
                    m_sBAD_WS_SUFFIX & Format(yDataWshCounter, "00")
                bBADDataWsCreated = True
            End If
            With wsBADData.Cells(x, y)
                .Interior.ColorIndex = 1
                .Font.ColorIndex = 2
            End With
            GoTo DoneWithCol
        End If
        '*******************
        '*******************
        'If HERE then we are going to create a CART "row"!!!...
        'If g_bStepThruMode Then Stop
        Dim tColDimenVals() As m_TypDimNmVal
        ReDim tColDimenVals(0 To yColDimens - 1)
        'Populate our array of column dimensions...
        For m = 0 To yColDimens - 1
            lColDimenRow = tColDimenRows(m).RowCol
            sDimen = Trim(tColDimenRows(m).nm)
            Select Case sDimen
                Case g_sDIMEN_DATA_SRC
                    tColDimenVals(m).SLRCol = g_yCOL_DIMEN_DATA_SRC
                    tColDimenVals(m).Val = _
                            vaDataSrc(1, 1 + y - lDataColFirst)
                Case g_sDIMEN_FUNC
                    tColDimenVals(m).SLRCol = g_yCOL_DIMEN_FUNC
                    tColDimenVals(m).Val = _
                            vaFunction(1, 1 + y - lDataColFirst)
                Case g_sDIMEN_LDGR
                    tColDimenVals(m).SLRCol = g_yCOL_DIMEN_LDGR
                    tColDimenVals(m).Val = _
                            vaLedger(1, 1 + y - lDataColFirst)
                Case g_sDIMEN_LGL_ENT
                    tColDimenVals(m).SLRCol = g_yCOL_DIMEN_LGL_ENT
                    tColDimenVals(m).Val = _
                            vaLglEnt(1, 1 + y - lDataColFirst)
                Case g_sDIMEN_MGD_BUS
                    tColDimenVals(m).SLRCol = g_yCOL_DIMEN_MGD_BUS
                    tColDimenVals(m).Val = _
                            vaMgdBus(1, 1 + y - lDataColFirst)
                Case g_sDIMEN_MV_ACCT
                    tColDimenVals(m).SLRCol = g_yCOL_DIMEN_MV_ACCT
                    tColDimenVals(m).Val = _
                            vaMVAcct(1, 1 + y - lDataColFirst)
                Case g_sDIMEN_OPER_LNS
```

```
                tColDimenVals(m).SLRCol = g_yCOL_DIMEN_OPER_LNS
                tColDimenVals(m).Val = _
                        vaOpLines(1, 1 + y - lDataColFirst)
            Case g_sDIMEN_ORIG_GEOG
                tColDimenVals(m).SLRCol = g_yCOL_DIMEN_ORIG_GEOG
                tColDimenVals(m).Val = _
                        vaOrigGeog(1, 1 + y - lDataColFirst)
            Case g_sDIMEN_PER
                tColDimenVals(m).SLRCol = g_yCOL_DIMEN_PER
                tColDimenVals(m).Val = _
                        vaPeriod(1, 1 + y - lDataColFirst)
            Case g_sDIMEN_PROD
                tColDimenVals(m).SLRCol = g_yCOL_DIMEN_PROD
                tColDimenVals(m).Val = _
                        vaProduct(1, 1 + y - lDataColFirst)
            Case g_sDIMEN_SCENAR
                tColDimenVals(m).SLRCol = g_yCOL_DIMEN_SCENAR
                tColDimenVals(m).Val = _
                        vaScenario(1, 1 + y - lDataColFirst)
            Case g_sDIMEN_YR
                tColDimenVals(m).SLRCol = g_yCOL_DIMEN_YR
                tColDimenVals(m).Val = _
                        vaYear(1, 1 + y - lDataColFirst)
        End Select
    Next m
    '******** PUSH DATA INTO A NEW ROW INTO OUR CART ARRAY...
    'If g_bStepThruMode Then Stop
    'Populate the metadata...
    vaCARTWshData(lCARTDataRow, m_yCART_WSHTYPE_COL) = sWshType
    vaCARTWshData(lCARTDataRow, m_yCART_WSH_COL) = sCurrDataWsh
    vaCARTWshData(lCARTDataRow, m_yCART_CELL_COL) = _
            wsSrcData.Cells(x, y).Address
    'Populate our file dimensions (if they exist!!)...
    If yFileDimens > 0 Then
        For m = 0 To yFileDimens - 1
            lCARTDataCol = tFileDimenVals(m).SLRCol + m_yCART_META_COLS
            vaCARTWshData(lCARTDataRow, lCARTDataCol) = _
                    tFileDimenVals(m).Val
        Next m
    End If
    'Populate our wsh dimensions (if they exist!!)...
    If yWshTypeDimens > 0 Then
        For m = 0 To yWshTypeDimens - 1
            lCARTDataCol = tWshTypeDimenVals(m).SLRCol + m_yCART_META_COLS
            vaCARTWshData(lCARTDataRow, lCARTDataCol) = _
                    tWshTypeDimenVals(m).Val
        Next m
    End If
    'Row dimensions...
    For m = 0 To yRowDimens - 1
        lCARTDataCol = tRowDimenVals(m).SLRCol + m_yCART_META_COLS
        vaCARTWshData(lCARTDataRow, lCARTDataCol) = _
                tRowDimenVals(m).Val
    Next m
    'Col dimensions...
    For m = 0 To yColDimens - 1
        lCARTDataCol = tColDimenVals(m).SLRCol + m_yCART_META_COLS
        vaCARTWshData(lCARTDataRow, lCARTDataCol) = _
                tColDimenVals(m).Val
    Next m
    'If g_bStepThruMode Then Stop
    'Amount...
    lCARTDataCol = g_yCOL_DIMEN_USD_AMMT + m_yCART_META_COLS
    dRawAmt = Val(vaAmts(1 + x - lDataRowFirst, _
            1 + y - lDataColFirst))
    If lScale > 1 Then
        cAdjAmt = dRawAmt * lScale
    Else
        cAdjAmt = dRawAmt
    End If
    vaCARTWshData(lCARTDataRow, lCARTDataCol) = cAdjAmt
```

```
FileHandler - 35

DoneWithCol:
                lCARTDataRow = lCARTDataRow + 1
            Next y
DoneWithRow:
        Next x
    'Push data into the CART table...
    If g_bStepThruMode Then Stop
    Dim rngCARTWshImport As Range

    Set rngCARTWshImport = _
            WshGoodData.Range(WshGoodData.Cells(lCARTRowNext, m_yCART_WSHTYPE_COL), _
                   WshGoodData.Cells(lCARTRowNext + lCARTWshRows - 1, _
                   m_yCART_WSHTYPE_COL + yCARTTtlCols - 1))
    rngCARTWshImport.Value2 = vaCARTWshData
    lCARTRowNext = lCARTRowNext + lCARTWshRows
DoneWithWsh:
    Next j
Next i

If g_bStepThruMode Then Stop
'Delete blank rows (if necessary) from the imported data...
Dim rngDataBodyOrig As Range
Dim rngDataBodyAmtCol As Range
WshGoodData.Activate '...just for yucks
Set rngDataBodyOrig = WshGoodData.UsedRange
Set rngDataBodyAmtCol = _
        rngDataBodyOrig.Columns(g_yCOL_DIMEN_USD_AMMT + m_yCART_META_COLS)
If WorksheetFunction.CountBlank(rngDataBodyAmtCol) > 0 Then
    Dim rngEmptyDataRows As Range
    Set rngEmptyDataRows = _
            rngDataBodyAmtCol.SpecialCells(xlCellTypeBlanks).EntireRow
    rngEmptyDataRows.Delete xlShiftUp
End If

'Do a "Ctrl-Home"...
Cells(ActiveWindow.SplitRow + 1, ActiveWindow.SplitColumn + 1).Select

If g_bStepThruMode Then Stop
'Create name for this workbook...
If g_bStepThruMode Then Stop
Dim sDateStamp As String
'chr(46) = period (".")
'Strip off file extension...
g_sSrcWbkNm = Left(g_sSrcWbkNm, InStrRev(g_sSrcWbkNm, Chr(46)) - 1) '...strip out
g_sSrcWbkNm = g_sSrcWbkNm & sCART_FILE_SUFFIX
sDateStamp = Str(Year(Now)) & Format(Month(Now), "00") & Format(Day(Now), "00")
g_sSrcWbkNm = g_sSrcWbkNm & sDateStamp
ThisWorkbook.SaveAs g_sSrcWbkNm

ThisWorkbook.Activate
WshGoodData.Activate

g_cMsgBoxHandler.SetDataImportCompleteMsg
g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbOKOnly

RibbonHandler.SetNextStageAndResetRibbon projNxtStgExportToCSV

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Application.DisplayAlerts = True
Application.ScreenUpdating = True
Application.StatusBar = vbNullString
bProcedurePreviouslyRun = True
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
```

```
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Public Sub CreateCSVFile(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "CreateCSVFile()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Const sCSV_TEMP_NM As String = "CART_csv_temp"
Dim wbCARTForCSV As Workbook
Dim wsCSVFile As Worksheet
Dim rngMetaDataCols As Range
Dim sCSVTempFileFullNm As String
Dim sCARTWbkNm As String
Dim ansCreateCSV As VbMsgBoxResult
Dim bOldCsvTempFileExists As Boolean

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
If g_bStepThruMode Then Stop

'Double check with user...
g_cMsgBoxHandler.AskExportToCSVQuest
ansCreateCSV = g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbYesNo)
If ansCreateCSV = vbNo Then GoTo ExitPoint

'See if a CART csv temp file is kicking around...
sCSVTempFileFullNm = ThisWorkbook.Path & Chr(92) & sCSV_TEMP_NM & ".xlsx"
bOldCsvTempFileExists = Utilities.DoesFolderOrFileExist(sCSVTempFileFullNm)
If bOldCsvTempFileExists Then Kill sCSVTempFileFullNm

'Create a new workbook to hold our csv data...
Set wbCARTForCSV = Workbooks.Add
wbCARTForCSV.SaveAs sCSV_TEMP_NM, xlOpenXMLWorkbook
WshGoodData.Copy before:=wbCARTForCSV.Worksheets(1)
Set wsCSVFile = wbCARTForCSV.Worksheets(WshGoodData.name)
'Delete the meta-data columns...
Set rngMetaDataCols = _
        wsCSVFile.Range(Cells(1, 1), Cells(1, m_yCART_META_COLS)).EntireColumn
rngMetaDataCols.Delete xlShiftToLeft
'Close and then reopen the file...
wbCARTForCSV.Close savechanges:=True
Workbooks.Open sCSV_TEMP_NM & ".xlsx"
'Reset, since we closed the workbook...
Set wbCARTForCSV = Workbooks(sCSV_TEMP_NM & ".xlsx")
Set wsCSVFile = wbCARTForCSV.Worksheets(WshGoodData.name)
wsCSVFile.Activate
'Get CSV file name (it will be same as this workbook's new name)...
'chr(46) = period (".")
'Strip off file extension...
sCARTWbkNm = ThisWorkbook.name
'My origninal line was this:
'sCARTWbkNm = Left(ThisWorkbook.name, InStrRev(ThisWorkbook.name, Chr(46)) - 1)
'However, in one test case "ThisWorkbook.name" yielded the name sans file _
extension.  So...
sCARTWbkNm = Replace(sCARTWbkNm, ".xlsm", vbNullString)

'It looks as though periods in the file name give us problems when saving _
```

```
FileHandler - 37

*.csv files, so [chr(95) = underscore ("_")]...
sCARTWbkNm = Replace(sCARTWbkNm, Chr(46), Chr(95))

'Now save this as a csv file...
wbCARTForCSV.SaveAs Filename:=sCARTWbkNm, FileFormat:=xlCSV
Workbooks(sCARTWbkNm & ".csv").Close savechanges:=False

g_cMsgBoxHandler.SetExportSuccesfulMsg
g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbInformation

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub



'================================================================================
'================================================================================
'================================================================================
'================================================================================
'================================================================================
'================================================================================
```

```
'.....................................................................................
'.....................................................................................
'.......        Developer:              William H. White (consultant)
'.......        With:                   TEKsystems Inc.
'.......                                www.teksystems.com
'.......        For:                    AIG
'.......                                Financial Information Systems
'.......                                1 NY Plaza, 15th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         8/20/2013
'.......        Proj finished:          11/21/2013
'.......        Proj revised:           1/15/2014
'.....................................................................................
'.....................................................................................



'*************************************************************************************
'*************************************************************************************
'*************************  CLASS-LEVEL DECLARATIONS  ************************
'*************************************************************************************
'*************************************************************************************
Option Explicit
Option Compare Binary '...use cAsE senSITive string comparisons




'*************************************************************************************
'*************************************************************************************
'*******************************  PRIVATE MEMBERS  *******************************
'*************************************************************************************
'*************************************************************************************

'***************************************
'MODULE/CLASS-WIDE COLLECTIONS
'***************************************
Private m_collWshTypeDimForms As WshTypeDimensForms


'***************************************
'MODULE/CLASS-WIDE FORMS
'***************************************
Private m_frmFileMandDimens As frmWbkMandDimens
Private m_frmImportWshs As frmWshsToImport
Private m_frmAddRemWshTypes As frmAddRemWshTypes
Private m_frmWshTypesWshAssoc As frmTypeWshAssoc
Private m_frmWshTypDimens As frmWshTypeDimens
Private m_frmWshTypePicker As frmWshTypePicker


'***************************************
'MODULE/CLASS-WIDE CONSTANTS
'***************************************
'Strings...
Private Const m_sMODULE_NAME As String = "FormsHandler"


'***************************************
'MODULE/CLASS-WIDE VARIABLES
'***************************************
'Booleans...
'NOTE:  These variables help us handle cases where the user is opening a _
virgin workbook but, after filling out and submitting data on a form, _
elects to go back and change something on that form.  (In effect, for that _
form we no longer want to treat the source wbk as virgin.)  We go ahead _
and set all these to TRUE when we are dealing with a previously-opened wbk...
```

```
Private m_bFileDimensPreviouslySet As Boolean
Private m_bDataWshsPreviouslySet As Boolean
Private m_bWshTypesPreviouslySet As Boolean
Private m_bWshTypeWshNmPreviouslySet As Boolean




'============================================================
'======================  EVENTS  ============================
'============================================================
'=============================================================================
Private Sub Class_Initialize()
'[VBA QUIRK:  Errors in Intialize, Activate, and Terminate event handlers _
cannot be handled by procedures which create or destroy instances of the _
class. If any such error-handling IS required, create a separate public _
procedure - e.g., the Initialize procedure, below.]
Set m_collWshTypeDimForms = New WshTypeDimensForms
End Sub
'=============================================================================
Private Sub Class_Terminate()
'[VBA QUIRK:  Errors in Intialize, Activate, and Terminate event handlers _
cannot be handled by procedures which create or destroy instances of the _
class. If any such error-handling IS required, create a separate public _
procedure - e.g., the Initialize procedure, below.]

'Since there's really nothing useful an error-handler can do at this point...
On Error Resume Next
'clean-up code...
End Sub




'============================================================
'==================  (Private) FUNCTIONS  ==================
'============================================================
'=============================================================================
Private Function WbkSourceRangeLargerThanOneCell( _
    ByVal RangeAddress As String, _
    ByRef RanOkRESULT As Boolean) As Boolean
'Error-handling declarations...
Const sSOURCE As String = "WbkSourceRangeLargerThanOneCell()"
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
'''Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim rngToTest As Range
Dim bLargerThanCell As Boolean '...false

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Set rngToTest = Range(RangeAddress)
If rngToTest.Columns.Count > 1 Or rngToTest.Rows.Count > 1 Then _
    bLargerThanCell = True
WbkSourceRangeLargerThanOneCell = bLargerThanCell

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Function

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
```

```
FormsHandler - 3

                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bFUNCTION_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Function
'================================================================================
Private Function CreateArraySourcWbkWshNames( _
    ByVal NmbrPossibleDataWorksheets As Integer, _
    ByRef RanOkRESULT As Boolean) As String()
'Error-handling declarations...
Const sSOURCE As String = "CreateArraySourcWbkWshNames()"
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
'''Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim ws As Worksheet
Dim sWshNms() As String
Dim sWsh As String
Dim i As Integer

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
ReDim sWshNms(0 To NmbrPossibleDataWorksheets - 1) As String
For Each ws In g_wbSrcData.Worksheets
    'Add wsh, so long as it isn't one of our added worksheets...
    If StrComp(ws.name, g_sWSH_DIMEN_LOCS) <> 0 And _
        StrComp(ws.name, g_sWSH_VAL_LISTS) Then
            sWshNms(i) = ws.name
            i = i + 1
    End If
Next ws
CreateArraySourcWbkWshNames = sWshNms()

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Function

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bFUNCTION_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Function
'================================================================================
Private Function AnyInvalidRangeReferences( _
    ByVal DimenScope As ProjDimenScope, _
    ByRef InvalidFieldRESULT As String, _
    Optional WshTypeNm As String) As Boolean
Dim refControl As RefEdit.RefEdit
Dim objRefEdit As Object
Dim objFormToCheck As Object
```

```
FormsHandler - 4

Dim sRngAddr As String
Dim vrtCellValue As Variant
Dim bInvalidRefs As Boolean '...default = false

'We don't run this function through our central error handler.  We just want _
to see if one line in the code throws an error...
On Error GoTo ErrorHandler

'If g_bStepThruMode Then Stop
If g_bMandDimensOnly Then
    Select Case DimenScope
        Case projScopeFile
            For Each refControl In m_frmFileMandDimens.RefEditsByCtrlNm
                If StrComp(Trim(refControl.value), vbNullString) <> 0 And _
                        refControl.enabled Then
                    'If g_bStepThruMode Then Stop
                    sRngAddr = refControl.value
                    'We don't care here about actual value, we just want _
                    to see if this statement throws an error...
                    vrtCellValue = Range(sRngAddr)
                End If
            Next refControl
        Case projScopeWshType
            For Each refControl In _
                    m_collWshTypeDimForms.Item(WshTypeNm).AllRefEditsByCtrlNm
                If StrComp(Trim(refControl.value), vbNullString) <> 0 And _
                        refControl.enabled Then
                    'If g_bStepThruMode Then Stop
                    sRngAddr = refControl.value
                    vrtCellValue = Range(sRngAddr)
                End If
            Next refControl
        Case Else
            Err.Raise g_lERR_INAPPROP_DIMEN_SCOPE
    End Select
Else '...ALL dimensions form(s)

End If

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
AnyInvalidRangeReferences = bInvalidRefs
Exit Function

ErrorHandler:
    bInvalidRefs = True
    Err.Clear
    InvalidFieldRESULT = refControl.Tag
    Resume ExitPoint
End Function
'================================================================================
Private Function AnyMultiCellRangeReferences( _
    ByVal DimenScope As ProjDimenScope, _
    ByRef InvalidFieldRESULT As String, _
    ByRef RanOkRESULT As Boolean, _
    Optional WshTypeNm As String) As Boolean
'Error-handling declarations...
Const sSOURCE As String = "AnyMultiCellRangeReferences()"
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim refCtrl As RefEdit.RefEdit
Dim sRngAddr As String
Dim bMultiCellRefs As Boolean

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'If g_bStepThruMode Then Stop
```

```
FormsHandler - 5

'CODE HERE...
If g_bMandDimensOnly Then
    Select Case DimenScope
        Case projScopeFile
            For Each refCtrl In m_frmFileMandDimens.RefEditsByCtrlNm
                If StrComp(Trim(refCtrl), vbNullString) <> 0 And _
                        refCtrl.enabled Then
                    sRngAddr = refCtrl '...default property = value
                    If Range(sRngAddr).Columns.Count > 1 Then
                        bMultiCellRefs = True
                        InvalidFieldRESULT = refCtrl.Tag
                        GoTo ExitPoint
                    End If
                    If Range(sRngAddr).Rows.Count > 1 Then
                        bMultiCellRefs = True
                        InvalidFieldRESULT = refCtrl.Tag
                        GoTo ExitPoint
                    End If
                End If
            Next refCtrl
        Case projScopeWshType
            For Each refCtrl In _
                    m_collWshTypeDimForms(WshTypeNm).WshDimenRefEditsByDimenNm
                If StrComp(Trim(refCtrl), vbNullString) <> 0 And _
                        refCtrl.enabled Then
                    sRngAddr = refCtrl '...default property = value
                    If Range(sRngAddr).Columns.Count > 1 Then
                        bMultiCellRefs = True
                        InvalidFieldRESULT = refCtrl.Tag
                        GoTo ExitPoint
                    End If
                    If Range(sRngAddr).Rows.Count > 1 Then
                        bMultiCellRefs = True
                        InvalidFieldRESULT = refCtrl.Tag
                        GoTo ExitPoint
                    End If
                End If
            Next refCtrl
        Case Else
            Err.Raise g_lERR_INAPPROP_DIMEN_SCOPE
    End Select
Else '...using ALL dimensions form


    Select Case DimenScope
        Case projScopeFile
'            For Each refCtrl In m_frmFileAllDimen.RefEditsByCtrlNm
'                If StrComp(Trim(refCtrl), vbNullString) <> 0 And _
'                        refCtrl.enabled Then
'                    sRngAddr = refCtrl '...default property = value
'                    If Range(sRngAddr).Columns.Count > 1 Then
'                        bMultiCellRefs = True
'                        InvalidFieldRESULT = refCtrl.Tag
'                        GoTo ExitPoint
'                    End If
'                    If Range(sRngAddr).Rows.Count > 1 Then
'                        bMultiCellRefs = True
'                        InvalidFieldRESULT = refCtrl.Tag
'                        GoTo ExitPoint
'                    End If
'                End If
'            Next refCtrl
        Case Else
            'do nothing...
    End Select
End If

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
AnyMultiCellRangeReferences = bMultiCellRefs
```

```
FormsHandler - 6

Exit Function

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bFUNCTION_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Function
'================================================================================
Private Function AnyDupeDimensionDefinitions( _
    ByVal WshTypeNm As String, _
    ByRef DupedDimenDefRESULT As String, _
    ByRef RanOkRESULT As Boolean) As Boolean
'Error-handling declarations...
Const sSOURCE As String = "AnyDupeDimensionDefinitions()"
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim refWsh As RefEdit.RefEdit
Dim refRow As RefEdit.RefEdit
Dim refCol As RefEdit.RefEdit
Dim sDimenNm As String
Dim bDupeExists As Boolean

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
'Loop through the Wsh-level dimensions.  For each that has been declared _
see if any declarations exist at the row or col level...
For Each refWsh In _
        m_collWshTypeDimForms.Item(WshTypeNm).WshDimenRefEditsByDimenNm
    If StrComp(Trim(refWsh), vbNullString) <> 0 Then
        sDimenNm = Trim(refWsh.Tag)
        'Check against row dimension...
        Set refRow = m_collWshTypeDimForms(WshTypeNm). _
                RowDimenRefEditsByDimenNm.Item(sDimenNm)
        If StrComp(Trim(refRow), vbNullString) <> 0 Then
            bDupeExists = True
            DupedDimenDefRESULT = sDimenNm
            GoTo ExitPoint
        End If
        'Check against row dimension...
        Set refCol = m_collWshTypeDimForms(WshTypeNm). _
                ColDimenRefEditsByDimenNm.Item(sDimenNm)
        If StrComp(Trim(refCol), vbNullString) <> 0 Then
            bDupeExists = True
            DupedDimenDefRESULT = sDimenNm
            GoTo ExitPoint
        End If
    End If
Next refWsh

'Now check row dimens against col dimens...
For Each refRow In _
        m_collWshTypeDimForms(WshTypeNm).RowDimenRefEditsByDimenNm
    If StrComp(Trim(refRow), vbNullString) Then
        sDimenNm = Trim(refRow.Tag)
        Set refCol = m_collWshTypeDimForms(WshTypeNm). _
                ColDimenRefEditsByDimenNm.Item(sDimenNm)
```

```
        If StrComp(Trim(refCol), vbNullString) <> 0 Then
            bDupeExists = True
            DupedDimenDefRESULT = sDimenNm
            GoTo ExitPoint
        End If
    End If
Next refRow

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
AnyDupeDimensionDefinitions = bDupeExists
Exit Function

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bFUNCTION_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Function
'================================================================================
Private Function AnyInvalidRowColRangeReferences( _
    ByRef DimScopeRESULT As ProjDimenScope, _
    ByRef InvalidRowDimenRESULT As String, _
    ByRef invalidColDimenRESULT As String, _
    ByRef RanOkRESULT As Boolean, _
    Optional WshTypeNm As String) As Boolean
'Error-handling declarations...
Const sSOURCE As String = "AnyInvalidRowColRangeReferences()"
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim refCtrl As Object
Dim sRngAddr As String
Dim bMultiRowsCols As Boolean

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
If g_bMandDimensOnly Then
    'Check Row Dimensions first...
    For Each refCtrl In _
            m_collWshTypeDimForms.Item(WshTypeNm).RowDimenRefEditsByDimenNm
        If StrComp(Trim(refCtrl), vbNullString) <> 0 And _
                refCtrl.enabled Then
            sRngAddr = refCtrl '...default property = value
            If Range(sRngAddr).Columns.Count > 1 Then
                bMultiRowsCols = True
                InvalidRowDimenRESULT = refCtrl.Tag
                DimScopeRESULT = projScopeRow
                GoTo ExitPoint
            End If
        End If
    Next refCtrl
    'Check Column Dimensions...
    For Each refCtrl In _
            m_collWshTypeDimForms.Item(WshTypeNm).ColDimenRefEditsByDimenNm
        If StrComp(Trim(refCtrl), vbNullString) <> 0 And _
                refCtrl.enabled Then
```

```
            sRngAddr = refCtrl '...default property = value
            If Range(sRngAddr).Rows.Count > 1 Then
                bMultiRowsCols = True
                InvalidRowDimenRESULT = refCtrl.Tag
                DimScopeRESULT = projScopeCol
                GoTo ExitPoint
            End If
        End If
    Next refCtrl

Else '...using ALL dimensions form

End If

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
AnyInvalidRowColRangeReferences = bMultiRowsCols
Exit Function

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bFUNCTION_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Function
'================================================================================
Private Function FormatRangeAddress( _
    ByVal DimenScope As ProjDimenScope, _
    ByVal RawRngAddr As String, _
    ByRef RanOkRESULT As Boolean) As String
'Error-handling declarations...
Const sSOURCE As String = "FormatRangeAddress()"
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
'''Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim sFormattedAddr As String

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Select Case DimenScope
    Case projScopeFile
'        'Our file range references are losing their leading apostrophe, so add it back
'        'First, see if the range name contains an apostrophe, other than in the 1st character...
'        If InStr(2, RawRngAddr, Chr(39)) > 0 Then
'            If StrComp(Chr(39), Left(Trim(RawRngAddr), 1), vbTextCompare) <> 0 Then
'                RawRngAddr = Chr(39) & Trim(RawRngAddr)
'            End If
'        End If
        sFormattedAddr = RawRngAddr
    Case projScopeWsh
        sFormattedAddr = Range(RawRngAddr).AddressLocal
    Case projScopeRow
        sFormattedAddr = Range(RawRngAddr).EntireColumn.AddressLocal
    Case projScopeCol
        sFormattedAddr = Range(RawRngAddr).EntireRow.AddressLocal
End Select
```

```
FormsHandler - 9

FormatRangeAddress = sFormattedAddr

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Function

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bFUNCTION_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Function



'============================================================
'==================  (Private) PROCEDURES  ==================
'============================================================
'================================================================================
Private Sub ClearAndRepopulateDimenTblFmWbkDimenForm( _
        ByVal AmtScale As ProjAmtScale, _
        ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "ClearAndRepopulateDimenTblFmWbkDimenForm()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim objControl As Object
Dim txtCtrl As MSForms.TextBox
Dim refCtrl As RefEdit.RefEdit
Dim wsDimenLocs As Worksheet
Dim rngFileDimens As Range
Dim tblDimLocs As ListObject
Dim sRefAddress As String
Dim sScale As String

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'Clear out Dimen Table values...
Set wsDimenLocs = g_wbSrcData.Worksheets(g_sWSH_DIMEN_LOCS)
Set tblDimLocs = wsDimenLocs.ListObjects("tblDimenLocs")
Set rngFileDimens = Application.Union( _
        tblDimLocs.DataBodyRange.Columns(g_yDIM_TBL_FD_HC_COL), _
        tblDimLocs.DataBodyRange.Columns(g_yDIM_TBL_FD_RR_COL))
rngFileDimens.ClearContents

'If g_bStepThruMode Then Stop
'Loop through RefEdit controls...
Dim sRngAddr As String
For Each refCtrl In m_frmFileMandDimens.RefEditsByCtrlNm
    If refCtrl.enabled And StrComp(Trim(refCtrl), vbNullString) <> 0 Then
        'Adjust Wsh Type, Row, and Column dimensions for desired format...
        sRngAddr = FormatRangeAddress( _
                DimenScope:=projScopeFile, _
                RawRngAddr:=refCtrl, _
                RanOkRESULT:=bCalledProcedureRanOk)
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
```

```
FormsHandler - 10

        g_cDimHandler.PopulateDimenTableWithValue _
                DimenScope:=projScopeFile, _
                TextInput:=False, _
                DimenName:=refCtrl.Tag, _
                ValueToStore:=sRngAddr, _
                RanOkRESULT:=bCalledProcedureRanOk
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    End If
Next refCtrl

'Loop through text box entries...
'If g_bStepThruMode Then Stop
For Each txtCtrl In m_frmFileMandDimens.TextBoxesByCtrlNm
    If txtCtrl.enabled And StrComp(Trim(txtCtrl), vbNullString) <> 0 Then
        'If g_bStepThruMode Then Stop
        g_cDimHandler.PopulateDimenTableWithValue _
                DimenScope:=projScopeFile, _
                TextInput:=True, _
                DimenName:=txtCtrl.Tag, _
                ValueToStore:=txtCtrl, _
                RanOkRESULT:=bCalledProcedureRanOk
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    End If
Next txtCtrl

If g_bStepThruMode Then Stop
'Handle amount units combo box...
Select Case AmtScale
    Case projScaleNone
        sScale = g_sSCALE_NONE
    Case projScaleThous
        sScale = g_sSCALE_THOUS
    Case projScaleMills
        sScale = g_sSCALE_MILLS
    Case projScaleBills
        sScale = g_sSCALE_BILLS
    Case Else
        'Do nothing... sScale remains vbnullstring
End Select

If StrComp(sScale, vbNullString) <> 0 Then
    g_cDimHandler.PopulateDimenTableWithValue _
            DimenScope:=projScopeFile, _
            TextInput:=True, _
            DimenName:=g_sDIMEN_SCALE, _
            ValueToStore:=sScale, _
            RanOkRESULT:=bCalledProcedureRanOk
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
End If

'If g_bStepThruMode Then Stop

''''If/when Amount Type is ever enabled...
'''If frmPageSheetDimens.lstAmtType.Enabled Then
'''    Dim enumAmtType As ProjAmtType
'''    Dim sAmtType As String
'''    Dim i As Byte
'''    'Get user's choice...
'''    For i = 1 To frmPageSheetDimens.lstAmtType.ListCount
'''        If frmPageSheetDimens.lstAmtType.Selected(i) Then
'''            sAmtType = frmPageSheetDimens.lstAmtType.ListIndex(i)
'''            i = frmPageSheetDimens.lstAmtType.ListCount
'''        End If
'''    Next i
'''
'''    'Set our enum variable...
'''    Select Case sAmtType
'''        Case "USD"
'''            enumAmtType = projAmtUSD
'''        Case "Transaction"
'''            enumAmtType = projAmtTrans
'''
```

```
FormsHandler - 11

'''        Case "Local"
'''            enumAmtType = projAmtLocal
'''        Case "Functional"
'''            enumAmtType = projAmtFunc
'''    End Select
'''
'''    'Specify type on appropriate dictionary...
'''    g_cDimHandler.AddFileDimenToCollAndDimenValueToDictDictionaryAmountType _
'''        DimensionScope:=projScopeFile, _
'''        AmountType:=enumAmtType, _
'''        RanOkRESULT:=bCalledProcedureRanOk
'''            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
'''End If

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Private Sub PopulateDimenTblFmWshTypeDimenForm( _
    ByVal WshTypeNm As String, _
    ByVal AmtScale As ProjAmtScale, _
    ByVal UndefinedMandDimens As Boolean, _
    ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "PopulateDimenTblFmWshTypeDimenForm()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim frmWTD As frmWshTypeDimens
Dim refCtrl As RefEdit.RefEdit
Dim wsDimenTbl As Worksheet
Dim rngDimenTbl As Range
Dim rngDimenTblDimenNmz As Range
Dim c As Range
Dim sDimLocTblNmzColAddr As String
Dim sDimLocTblWshTypeColAddr As String
Dim sRefAddress As String
Dim yDimenTblWshTypeCol As Byte
Dim yColOffsetNmToWshType As Byte
Dim bGetWshTypeDimen As Boolean
Dim bRngRefDefined As Boolean

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True
If g_bStepThruMode Then Stop

WshTypeNm = Trim(WshTypeNm)
Set frmWTD = Me.WshTypeDimensFormsColl.Item(WshTypeNm)
Set wsDimenTbl = g_wbSrcData.Worksheets(g_sWSH_DIMEN_LOCS)
sDimLocTblNmzColAddr = _
        "tblDimenLocs" & Chr(91) & g_sDIM_LOC_TBL_HDR_DIM_NM & Chr(93)
sDimLocTblWshTypeColAddr = "tblDimenLocs" & Chr(91) & Trim(WshTypeNm) & Chr(93)
```

```
yDimenTblWshTypeCol = wsDimenTbl.Range(sDimLocTblWshTypeColAddr).Column
yColOffsetNmToWshType = yDimenTblWshTypeCol - g_yDIM_TBL_NM_COL
Set rngDimenTblDimenNmz = wsDimenTbl.Range(sDimLocTblNmzColAddr)

'Clear out any existing table entries...
wsDimenTbl.Range(sDimLocTblWshTypeColAddr).ClearContents

'Loop through dimension names in DimenLocation table...
For Each c In rngDimenTblDimenNmz
    'First see if we even need to get WshType dimen...
    bGetWshTypeDimen = False '...reset
    bRngRefDefined = False '...reset
    If g_bMandDimensOnly And _
            c.Offset(0, g_yDT_COL_OFF_NM_2_MAND) And _
            StrComp(Trim(c.Offset(0, g_yDT_COL_OFF_NM_2_FD_HC)), _
                vbNullString) = 0 And _
            StrComp(Trim(c.Offset(0, g_yDT_COL_OFF_NM_2_FD_RR)), _
                vbNullString) = 0 Then bGetWshTypeDimen = True
    If Not g_bMandDimensOnly And _
            StrComp(Trim(c.Offset(0, g_yDT_COL_OFF_NM_2_FD_HC)), _
                vbNullString) = 0 And _
            StrComp(Trim(c.Offset(0, g_yDT_COL_OFF_NM_2_FD_RR)), _
                vbNullString) = 0 Then bGetWshTypeDimen = True
    If bGetWshTypeDimen Then
        'If g_bStepThruMode Then Stop
        If StrComp(Trim(c.value), g_sDIMEN_SCALE) = 0 Then
            'Grab scale info and put it in the sRefAddress variable, even _
            though it's not a RefAddress...
            If frmWTD.cmbScale.ListIndex = -1 Then
                sRefAddress = vbNullString
            Else
                sRefAddress = frmWTD.cmbScale.List(frmWTD.cmbScale.ListIndex, 0)
            End If
            bRngRefDefined = True
        Else
            'Check if dimension is defined at the worksheet level...
            Set refCtrl = frmWTD.WshDimenRefEditsByDimenNm.Item(c)
            'Get as-is range address...
            sRefAddress = refCtrl.value
            If StrComp(sRefAddress, vbNullString) <> 0 Then
                'Convert to eliminate wsh part of address...
                sRefAddress = Range(sRefAddress).Address
                bRngRefDefined = True
            End If
        End If

        If Not bRngRefDefined Then
            'Check if dimension is defined at the row level...
            Set refCtrl = frmWTD.RowDimenRefEditsByDimenNm.Item(c)
            'Get as-is range address...
            sRefAddress = refCtrl.value
            If StrComp(sRefAddress, vbNullString) <> 0 Then
                'Convert to appropriate format (i.e., col and no wsh name)...
                sRefAddress = Range(sRefAddress).EntireColumn.Address
                bRngRefDefined = True
            End If
        End If

        If Not bRngRefDefined Then
            'Check if dimension is defined at the column level...
            Set refCtrl = frmWTD.ColDimenRefEditsByDimenNm.Item(c)
            'Get as-is range address...
            sRefAddress = refCtrl.value
            If StrComp(sRefAddress, vbNullString) <> 0 Then
                'Convert to appropriate format (i.e., row and no wsh name)...
                sRefAddress = Range(sRefAddress).EntireRow.Address
                bRngRefDefined = True
            End If
        End If
        'Populate the DimenLoc table...
        If bRngRefDefined Then c.Offset(0, yColOffsetNmToWshType) = sRefAddress
```

```
    End If
Next c
If g_bStepThruMode Then Stop


ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Private Sub ForOneWshTypePopWshTypeCollAndWshToWshTypeMap( _
        ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "ForOneWshTypePopWshTypeCollAndWshToWshTypeMap()"
Const bSUB_IS_ENTRY_PT As Boolean = False
'''Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wsValLists As Worksheet
Dim wsSrcData As Worksheet
Dim rngDataWshs As Range
Dim rngWshNmWshTypeTblHdrz As Range
Dim sSingleWshType As String
Dim sWshNm As String
Dim yDataWshs As Byte
Dim i As Byte

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
If g_bStepThruMode Then Stop
Set wsValLists = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
Set rngDataWshs = wsValLists.Range("vallstdodynWshsToImport")
Set rngWshNmWshTypeTblHdrz = wsValLists.Range("valhdrzWshNmTypeMap")
sSingleWshType = Trim(wsValLists.Range("vallstdodynWshTypes").Cells(1, 1))
yDataWshs = rngDataWshs.Rows.Count

For i = 1 To yDataWshs
    rngWshNmWshTypeTblHdrz.Offset(i, 0).Cells(1, g_yWSH_ASSOC_TBL_TYPE_COL) = _
            sSingleWshType
    sWshNm = rngDataWshs.Cells(i, 1)
    rngWshNmWshTypeTblHdrz.Offset(i, 0).Cells(1, g_yWSH_ASSOC_TBL_WSH_NM_COL) = _
            sWshNm
    Set wsSrcData = g_wbSrcData.Worksheets(sWshNm)
Next i

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
```

```
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'============================================================================
Private Sub CheckForAndHandleUserAddingDataWshs( _
        ByRef DataWshsInFormRESULT() As String, _
        ByRef UserAddedWshsRESULT As Boolean, _
        ByRef UserCxlsAddingWshsToSingleWshTypeRESULT As Boolean, _
        ByRef UserAddedNewWshsToSingleTypeRESULT As Boolean, _
        ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "CheckForAndHandleUserAddingDataWshs()"
Const bSUB_IS_ENTRY_PT As Boolean = False
'''Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wsValLists As Worksheet
Dim rngDataWshsInXL As Range
Dim rngWshTypeInfoTbl As Range
Dim rngWshNmWshTypeMapTbl As Range
Dim saNewDataWshsInForm() As String
Dim sDataWshInForm As String
Dim iDataWshsInForm As Integer
Dim iDataWshsInXL As Integer
Dim iNewWshs As Integer
Dim i As Integer
Dim j As Integer
Dim bSingleWshType As Boolean
Dim bNewWsh As Boolean

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Set wsValLists = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
Set rngDataWshsInXL = wsValLists.Range("vallstdodynWshsToImport")
Set rngWshTypeInfoTbl = wsValLists.Range("valtbldodynWshTypeInfo")
Set rngWshNmWshTypeMapTbl = wsValLists.Range("valtbldodynWshNmTypeMap")
iDataWshsInXL = rngDataWshsInXL.Rows.Count
iDataWshsInForm = UBound(DataWshsInFormRESULT) + 1
If rngWshTypeInfoTbl.Rows.Count = 1 Then bSingleWshType = True

'The range will always have at least one cell...
If iDataWshsInXL = 1 And IsEmpty(rngDataWshsInXL.Cells(1, 1)) Then
    UserAddedWshsRESULT = True
    'This is apparently a virgin wkbk, so...
    GoTo ExitPoint
End If

'If g_bStepThruMode Then Stop
'If here then user had previously defined wshs to import.  Now see if any _
worhseets in frmWshsToImport are new...
For i = 1 To iDataWshsInForm
    sDataWshInForm = Trim(DataWshsInFormRESULT(i - 1))
    bNewWsh = True '...assume this until we find otherwise
    For j = 1 To iDataWshsInXL
        If StrComp(sDataWshInForm, Trim(rngDataWshsInXL.Cells(j, 1)), _
                vbTextCompare) = 0 Then
            bNewWsh = False
            j = iDataWshsInXL '...break loop
```

```
        End If
    Next j
    'If g_bStepThruMode Then Stop
    If bNewWsh Then
        ReDim Preserve saNewDataWshsInForm(0 To iNewWshs)
        saNewDataWshsInForm(iNewWshs) = sDataWshInForm
        iNewWshs = iNewWshs + 1
    End If
Next i

'If g_bStepThruMode Then Stop
If iNewWshs = 0 Then GoTo ExitPoint

'If here we have new worksheets...
UserAddedWshsRESULT = True

'Prompt user, basing the prompt on number of worsheet types...
If bSingleWshType Then
    Dim ansAddWshsToSingleWshType As VbMsgBoxResult
    Dim sWshType As String
    sWshType = rngWshTypeInfoTbl.Cells(1, g_yWSH_TYPE_TBL_NM_COL)
    g_cMsgBoxHandler.AskAddNewDataWshsToExistingWshTypeQuest _
            sWshType, saNewDataWshsInForm
    ansAddWshsToSingleWshType = _
            g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbYesNoCancel + vbQuestion)
    If g_bStepThruMode Then Stop
    If ansAddWshsToSingleWshType = vbCancel Then _
            UserCxlsAddingWshsToSingleWshTypeRESULT = True
    If ansAddWshsToSingleWshType = vbYes Then _
            UserAddedNewWshsToSingleTypeRESULT = True
Else
    'Tell user s/he will have to assign worksheets to a worksheet type...
    g_cMsgBoxHandler.SetNewDataWshsAddedMsg saNewDataWshsInForm
    g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbInformation
End If

If g_bStepThruMode Then Stop
If UserAddedNewWshsToSingleTypeRESULT Then
    'NOTE:  We are adding data wsh names to the wsh type/wsh name mapping _
    table before we even add the data wsh names to the data wsh names table, _
    but this seems like the best place in the code to do this _
    little procedure...
    Dim rngNewDataRow As Range
    Dim sSingleWshType As String
    Dim saNewDataRow(1 To 1, 1 To 2) As String
    Dim yOffsetRows As Byte
    sSingleWshType = Trim(rngWshTypeInfoTbl.Cells(1, g_yWSH_TYPE_TBL_NM_COL))
    For i = 1 To iNewWshs
        yOffsetRows = rngWshNmWshTypeMapTbl.Rows.Count + i - 1
        '2 is a magic number, I know, I know...
        Set rngNewDataRow = rngWshNmWshTypeMapTbl.Resize(1, 2)
        Set rngNewDataRow = rngNewDataRow.Offset(yOffsetRows, 0)
        saNewDataRow(1, 1) = saNewDataWshsInForm(i - 1)
        saNewDataRow(1, 2) = sSingleWshType
        rngNewDataRow.Value2 = saNewDataRow
    Next i
    'Now resort the table...
    Set rngWshNmWshTypeMapTbl = wsValLists.Range("valtbldodynWshNmTypeMap")
    'Resize to capture header row...
    Set rngWshNmWshTypeMapTbl = _
            rngWshNmWshTypeMapTbl.Resize(rngWshNmWshTypeMapTbl.Rows.Count + 1)
    Set rngWshNmWshTypeMapTbl = rngWshNmWshTypeMapTbl.Offset(-1, 0)
    rngWshNmWshTypeMapTbl.Sort _
            key1:=rngWshNmWshTypeMapTbl.Columns(g_yWSH_ASSOC_TBL_TYPE_COL), _
            key2:=rngWshNmWshTypeMapTbl.Columns(g_yWSH_ASSOC_TBL_WSH_NM_COL), _
            Header:=xlYes, _
            Orientation:=xlSortColumns
End If

ExitPoint:
On Error Resume Next
```

```
FormsHandler - 16

'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
            ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'================================================================================
Private Sub CheckForAndHandleUserRemovingDataWshs( _
        ByRef SelectedDataWshsInForm() As String, _
        ByRef DeselectedWshs As Collection, _
        ByRef ToBeEmptyWshTypes As Collection, _
        ByRef UserSaysDeleteWshTypesRESULT As Boolean, _
        ByRef EveryWshTypeWillBeEmptyRESULT As Boolean, _
        ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "CheckForAndHandleUserRemovingDataWshs()"
Const bSUB_IS_ENTRY_PT As Boolean = False
'''Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wsValLists As Worksheet
Dim rngDataWshsInXL As Range
Dim rngWshTypeInfoTbl As Range
Dim rngWshTypeWshNmMap As Range
Dim sWshType As String
Dim sDataWshInXL As String
Dim iDataWshsInForm As Integer
Dim iDataWshsInXL As Integer
Dim iExistingWshTypes As Integer
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim bSingleWshType As Boolean
Dim bWshTypeToBeEmpty As Boolean
Dim bWshRemoved As Boolean

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Set wsValLists = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
Set rngDataWshsInXL = wsValLists.Range("vallstdodynWshsToImport")
Set rngWshTypeInfoTbl = wsValLists.Range("valtbldodynWshTypeInfo")
Set rngWshTypeWshNmMap = wsValLists.Range("valtbldodynWshNmTypeMap")
iDataWshsInXL = rngDataWshsInXL.Rows.Count
iExistingWshTypes = rngWshTypeInfoTbl.Rows.Count
iDataWshsInForm = UBound(SelectedDataWshsInForm) + 1
If rngWshTypeInfoTbl.Rows.Count = 1 Then bSingleWshType = True

'Bail out if we are in a virgin workbook...
If StrComp(Trim(rngDataWshsInXL.Cells(1, 1)), vbNullString) = 0 Then GoTo ExitPoint

'If g_bStepThruMode Then Stop
'Loop through each Wsh in our XL table and see if it is to be removed...
For i = 1 To iDataWshsInXL
    bWshRemoved = True '...assume this until we find otherwise
    sDataWshInXL = Trim(rngDataWshsInXL.Cells(i, 1))
    For j = 1 To iDataWshsInForm
```

```
        If StrComp(sDataWshInXL, Trim(SelectedDataWshsInForm(j - 1)), _
                vbTextCompare) = 0 Then
            bWshRemoved = False
            j = iDataWshsInForm '...break loop
        End If
    Next j
    If bWshRemoved Then _
        DeselectedWshs.Add Item:=sDataWshInXL, Key:=sDataWshInXL
Next i

If g_bStepThruMode Then Stop
If DeselectedWshs.Count = 0 Then GoTo ExitPoint
If DeselectedWshs.Count = iDataWshsInXL Then _
        EveryWshTypeWillBeEmptyRESULT = True

'******If here user is removing some worksheets...

'NOW figure out if any of our WshTypes will be empty!...
For i = 1 To iExistingWshTypes
    sWshType = Trim(rngWshTypeInfoTbl.Cells(i, g_yWSH_TYPE_TBL_NM_COL))
    bWshTypeToBeEmpty = True '...until we find otherwise
    'Now find each instance of this Wsh Type in the WshType/WshNm _
    mapping table...
    For j = 1 To iDataWshsInXL
        If StrComp(sWshType, Trim(rngWshTypeWshNmMap.Cells(j, _
                g_yWSH_ASSOC_TBL_TYPE_COL)), vbTextCompare) = 0 Then
            'Get a data worksheet associated with this worksheet type...
            sDataWshInXL = _
                    Trim(rngWshTypeWshNmMap.Cells(j, g_yWSH_ASSOC_TBL_WSH_NM_COL))
            'See if the data worksheet is among those the user will import...
            For k = 1 To iDataWshsInForm
                If StrComp(sDataWshInXL, SelectedDataWshsInForm(k - 1), _
                        vbTextCompare) = 0 Then
                    'If yes then the worksheet type won't be empty...
                    bWshTypeToBeEmpty = False
                    k = iDataWshsInForm '...break loop
                End If
            Next k
            'Once we find that our wsh type has a still-to-be-imported data _
            worksheet associated with it we can stop checking that _
            worksheet type...
            If Not bWshTypeToBeEmpty Then j = iDataWshsInXL '...break loop
        End If
    Next j
    'Now we've checked all instances of the worksheet type within the _
    WshType/WshName mapping table.  Handle the case of a worksheet type that _
    will no longer have data worksheets associated with it after the user _
    de-selects the data worksheets he plans to de-select...
    If bWshTypeToBeEmpty Then ToBeEmptyWshTypes.Add Item:=sWshType, Key:=sWshType
Next i

'There is no need in and of itself to inform the user that he has decided _
to de-select a worksheet as one which contains CART data.  However, we DO _
want to double-check with a user if after de-selecting worksheets that there _
will be one or more worksheet types with no associated data worksheets...
If ToBeEmptyWshTypes.Count = 0 Then GoTo ExitPoint

'Prompt user about...
Dim ansDeselectWshs As VbMsgBoxResult
g_cMsgBoxHandler.AskDeSelectDataWshsEvenThoughEmptyWshTypesWillBeDeletedQuest _
    DeselectedWshs, ToBeEmptyWshTypes
ansDeselectWshs = _
        g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbQuestion + vbYesNoCancel)
If ansDeselectWshs = vbYes Then UserSaysDeleteWshTypesRESULT = True

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
```

```
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
            ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'================================================================================
Private Sub AddALLWshType(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "AddALLWshType()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim objAllWshType As WshType
Dim wsDimenLocs As Worksheet
Dim wsValLists As Worksheet
Dim tblDimenLocs As ListObject
Dim lcoAllType As ListColumn
Dim rngWshTypeStatusTbl As Range
Dim rngNewWshTypeRow As Range

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Set wsDimenLocs = g_wbSrcData.Worksheets(g_sWSH_DIMEN_LOCS)
Set wsValLists = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
Set tblDimenLocs = wsDimenLocs.ListObjects("tblDimenLocs")
Set rngWshTypeStatusTbl = wsValLists.Range("valtblwhdynWshTypeInfo")

'Add a column in the DimenLocs table...
Set lcoAllType = tblDimenLocs.ListColumns.Add()
lcoAllType.name = g_sWSH_TYPE_ALL_PC
lcoAllType.DataBodyRange.EntireColumn.ColumnWidth = g_yCOL_WIDTH_DIMEN_LOCS

'Add a row in the WshType Status table...
Set rngNewWshTypeRow = rngWshTypeStatusTbl.Resize(1, 2)
Set rngNewWshTypeRow = _
        rngNewWshTypeRow.Offset(rngWshTypeStatusTbl.Rows.Count, 0)
rngNewWshTypeRow.Cells(1, g_yWSH_TYPE_TBL_NM_COL) = g_sWSH_TYPE_ALL_PC
rngNewWshTypeRow.Cells(1, g_yWSH_TYPE_TBL_DEFINED_COL) = False

'Add ALL type to DimenHandler's WorksheetTypes collection...
Set objAllWshType = g_cObjFactory.CreateWshType( _
        WshTypeName:=g_sWSH_TYPE_ALL_PC, _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
g_cDimHandler.WorksheetTypes.Add objAllWshType

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
            ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
```

```
            IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'============================================================================
Private Sub RemoveWshTypeFmXLValListsAndDimenLocTbl( _
        ByVal WshTypeNm As String, _
        ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "RemoveWshTypeFmXLValListsAndDimenLocTbl()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wsValLists As Worksheet
Dim wsDimLocs As Worksheet
Dim rngWshTypeInfoTblInclHdrz As Range
Dim rngWshTypeWshNmTblInclHdrz As Range
Dim rngRowToClear As Range
Dim rngEmptyWshTypeCol As Range
Dim i As Byte

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
WshTypeNm = Trim(WshTypeNm)
Set wsValLists = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
Set wsDimLocs = g_wbSrcData.Worksheets(g_sWSH_DIMEN_LOCS)
Set rngWshTypeInfoTblInclHdrz = wsValLists.Range("valtblwhdynWshTypeInfo")
Set rngWshTypeWshNmTblInclHdrz = wsValLists.Range("valtbldodynWshNmTypeMap")
Set rngWshTypeWshNmTblInclHdrz = _
        rngWshTypeWshNmTblInclHdrz.Resize( _
        rngWshTypeWshNmTblInclHdrz.Rows.Count + 1)
Set rngWshTypeWshNmTblInclHdrz = rngWshTypeWshNmTblInclHdrz.Offset(-1, 0)

'Clear wshtype from WshTypeInfo table...
For i = 2 To rngWshTypeInfoTblInclHdrz.Rows.Count
    If StrComp(WshTypeNm, _
            Trim(rngWshTypeInfoTblInclHdrz.Cells(i, g_yWSH_TYPE_TBL_NM_COL)), _
            vbTextCompare) = 0 Then
        Set rngRowToClear = rngWshTypeInfoTblInclHdrz.Rows(i)
        rngRowToClear.ClearContents
        'Since a wsh type appears only once in this table...
        i = rngWshTypeInfoTblInclHdrz.Rows.Count '...break loop
    End If
Next i

'Close up the empty rows...
rngWshTypeInfoTblInclHdrz.Sort _
        key1:=rngWshTypeInfoTblInclHdrz.Columns(g_yWSH_TYPE_TBL_NM_COL), _
        Header:=xlYes, _
        Orientation:=xlSortColumns

'Clear each instance of wshtype from WshType/WshNm mapping table...
For i = 2 To rngWshTypeWshNmTblInclHdrz.Rows.Count
    If StrComp(WshTypeNm, _
            Trim(rngWshTypeWshNmTblInclHdrz.Cells(i, g_yWSH_ASSOC_TBL_TYPE_COL)), _
            vbTextCompare) = 0 Then
        Set rngRowToClear = rngWshTypeWshNmTblInclHdrz.Rows(i)
        rngRowToClear.ClearContents
    End If
Next i

'Close up the empty rows...
rngWshTypeWshNmTblInclHdrz.Sort _
```

```
FormsHandler - 20

        key1:=rngWshTypeWshNmTblInclHdrz.Columns(g_yWSH_ASSOC_TBL_TYPE_COL), _
        key2:=rngWshTypeWshNmTblInclHdrz.Columns(g_yWSH_ASSOC_TBL_WSH_NM_COL), _
        Header:=xlYes, _
        Orientation:=xlSortColumns

If g_bStepThruMode Then Stop
'Remove the now-empty Wsh Type (column) from the Dimension Locs table...
Set rngEmptyWshTypeCol = _
        wsDimLocs.Range("tblDimenLocs" & Chr(91) & WshTypeNm & Chr(93))
rngEmptyWshTypeCol.EntireColumn.Delete xlShiftToLeft


ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
            ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'================================================================================
Private Sub RemoveDataWshFmWshNmWshTypeMapTbl( _
        ByVal ExDataWsh As String, _
        ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "RemoveDataWshFmWshNmWshTypeMapTbl()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wsValLists As Worksheet
Dim rngWshTypeWshNmTblInclHdrz As Range
Dim rngRowToClear As Range
Dim i As Byte

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
If g_bStepThruMode Then Stop
ExDataWsh = Trim(ExDataWsh)
Set wsValLists = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
Set rngWshTypeWshNmTblInclHdrz = wsValLists.Range("valtbldodynWshNmTypeMap")
Set rngWshTypeWshNmTblInclHdrz = _
        rngWshTypeWshNmTblInclHdrz.Resize( _
        rngWshTypeWshNmTblInclHdrz.Rows.Count + 1)
Set rngWshTypeWshNmTblInclHdrz = rngWshTypeWshNmTblInclHdrz.Offset(-1, 0)

'Clear the data worksheet from WshType/WshNm mapping table...
For i = 2 To rngWshTypeWshNmTblInclHdrz.Rows.Count
    If StrComp(ExDataWsh, _
            Trim(rngWshTypeWshNmTblInclHdrz.Cells(i, g_yWSH_ASSOC_TBL_WSH_NM_COL)), _
            vbTextCompare) = 0 Then
        Set rngRowToClear = rngWshTypeWshNmTblInclHdrz.Rows(i)
        rngRowToClear.ClearContents
        i = rngWshTypeWshNmTblInclHdrz.Rows.Count
    End If
Next i
```

```
FormsHandler - 21


'Close up the empty rows...
rngWshTypeWshNmTblInclHdrz.Sort _
        key1:=rngWshTypeWshNmTblInclHdrz.Columns(g_yWSH_ASSOC_TBL_TYPE_COL), _
        key2:=rngWshTypeWshNmTblInclHdrz.Columns(g_yWSH_ASSOC_TBL_WSH_NM_COL), _
        Header:=xlYes, _
        Orientation:=xlSortColumns


ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
            ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub



'***************************************************************************
'***************************************************************************
'*****************************  PUBLIC MEMBERS  ****************************
'***************************************************************************
'***************************************************************************

'=========================================================
'======================  PROPERTIES  =====================
'=========================================================
'===========================================================================
'Property WshTypeDimensFormsColl (read-only)...
Property Get WshTypeDimensFormsColl() As WshTypeDimensForms
    Set WshTypeDimensFormsColl = m_collWshTypeDimForms
End Property




'=========================================================
'==================  (Public) FUNCTIONS  =================
'=========================================================
'===========================================================================
Public Function IsFormOpen( _
        ByVal FormName As String, _
        ByRef RanOkRESULT As Boolean, _
        Optional WshTypeNm As String) As Boolean
'Error-handling declarations...
Const sSOURCE As String = "IsFormOpen()"
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim bFormOpen As Boolean

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
```

```
Select Case FormName
    Case frmWbkMandDimens.name
        If Not m_frmFileMandDimens Is Nothing Then bFormOpen = True
    Case frmWshsToImport.name
        If Not m_frmImportWshs Is Nothing Then bFormOpen = True
    Case frmAddRemWshTypes.name
        If Not m_frmAddRemWshTypes Is Nothing Then bFormOpen = True
    Case frmTypeWshAssoc.name
        If Not m_frmWshTypesWshAssoc Is Nothing Then bFormOpen = True
    Case frmWshTypeDimens.name
        'If count = 0 then answer is False...
        If Me.WshTypeDimensFormsColl.Count > 0 Then _
            bFormOpen = Me.WshTypeDimensFormsColl.ContainsItem(WshTypeNm)
    Case frmWshTypePicker.name
        If Not m_frmWshTypePicker Is Nothing Then bFormOpen = True
    Case Else
        'Bogus form name, so...
        Err.Raise g_lERR_MISMATCHED_WSHS
End Select

IsFormOpen = bFormOpen

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Function

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
            ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bFUNCTION_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Function



'============================================================
'==================  (Public) PROCEDURES  ==================
'============================================================
'================================================================================
Public Sub Initialize( _
    ByVal NextStg As ProjNextStage, _
    ByRef RanOkRESULT As Boolean)
'[NOTE:  Use this procedure whenever there is start-up code which might _
possibly fail.  Put such code here, which is a procedure which must be _
called expicitly, because, as noted above, errors which might occur in _
the Class_Initialize() method CANNOT be trapped by a calling procedure.]

'Error-handling declarations...
Const sSOURCE As String = "Initialize()"
Const bPROCEDURE_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until an error is encountered...
RanOkRESULT = True

If Not g_bWizardMode Then
    Select Case NextStg
        Case projNxtStgFileDimens
```

```
            'Do nothing - nothing previously set
        Case projNxtStgIDDataWshs
            m_bFileDimensPreviouslySet = True
        Case projNxtStgCreateWshTyps
            m_bFileDimensPreviouslySet = True
            m_bDataWshsPreviouslySet = True
        Case projNxtStgAssocWshTyps
            m_bFileDimensPreviouslySet = True
            m_bDataWshsPreviouslySet = True
            m_bWshTypesPreviouslySet = True
        Case projNxtStgWshTypeDimens
            m_bFileDimensPreviouslySet = True
            m_bDataWshsPreviouslySet = True
            m_bWshTypesPreviouslySet = True
            m_bWshTypeWshNmPreviouslySet = True
    End Select
End If

ExitPoint:
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bPROCEDURE_IS_ENTRY_PT
        If bRetraceErrorMode Then
                Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Public Sub OpenWbkDimenIDForm(ByRef RanOkRESULT As Boolean)
Const sSOURCE As String = "OpenWbkDimenIDForm()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Const sSTEP_FILE_TTL As String = "STEP 2 OF 8:  "
Const sFILE_DIMEN_FORM_TTL As String = _
    "File Dimensions for...  "
Const sWSH_DIMEN_FORM_TTL As String = "Dimensions for...  "
Dim objControl As Object
Dim objDimen As Dimension
Dim sFormCaption As String
Dim bFileDimensInXL As Boolean

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True
'If g_bStepThruMode Then Stop
'CODE HERE...
If g_bMandDimensOnly Then
    sFormCaption = sFILE_DIMEN_FORM_TTL & g_sSrcWbkNm
    If g_bWizardMode Then sFormCaption = sSTEP_FILE_TTL & sFormCaption
    Set m_frmFileMandDimens = _
        g_cObjFactory.CreateWbkMandDimensForm( _
                FormCaption:=sFormCaption, _
                RanOkRESULT:=bCalledProcedureRanOk)
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    If g_bStepThruMode Then Stop
    bFileDimensInXL = g_cDimHandler.AnyFileDimensDefined( _
            RanOkRESULT:=bCalledProcedureRanOk)
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    If bFileDimensInXL Then
        g_cDimHandler.PopulateWbkDimenFormFmSrcWbkDimenTbl _
                RanOkRESULT:=bCalledProcedureRanOk
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    End If
```

```
    m_frmFileMandDimens.Show
Else
    'If the project is ever revised to cover non-mandatory dimensions, _
    populating that form will go here...
End If

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Public Sub OpenWshTypeDimenIDForm( _
        ByVal OpenedFmAnotherWTDForm As Boolean, _
        ByRef RanOkRESULT As Boolean, _
        Optional WshTypeNm As String) 'Error-handling declarations...
Const sSOURCE As String = "OpenWshTypeDimenIDForm()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Const sSTEP_WshTypeNm_CAP As String = "STEP 6 OF 8:  "
Const sWSH_DIMEN_FORM_TTL As String = "Dimensions for...  "
Dim frmWTD As frmWshTypeDimens
Dim objDimen As Dimension
Dim WsType As WshType
Dim wsValLists As Worksheet
Dim rngdoWshTypesTbl As Range
Dim sFormCaption As String
Dim yWshTypesInTable As Byte
Dim i As Byte

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True
If g_bStepThruMode Then Stop

'CODE HERE...
'Set some variables...
WshTypeNm = Trim(WshTypeNm)
Set wsValLists = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
Set rngdoWshTypesTbl = wsValLists.Range("valtbldodynWshTypeInfo")
yWshTypesInTable = rngdoWshTypesTbl.Rows.Count


If yWshTypesInTable = 1 Then
    'This is an optional variable and would have been empty...
    WshTypeNm = rngdoWshTypesTbl.Cells(1, g_yWSH_TYPE_TBL_NM_COL)
End If

'Figure out the WshType we are going to use to open the form.  If the _
user clicked on our CART SL ribbon no Wsh type is supplied and _
(sigh...) we have to pick one for him...
If StrComp(WshTypeNm, vbNullString) = 0 Then
    'Start by grabbing the first WshType in the table...
    WshTypeNm = Trim(rngdoWshTypesTbl.Cells(1, g_yWSH_TYPE_TBL_NM_COL))
```

```
    'If there's only one WshType we're done.  Otherwise...
    If yWshTypesInTable > 1 Then
        'It only makes sense to change if there are undefined types. _
        If there are simply pick the first one...
        For i = 1 To yWshTypesInTable
            If Not rngdoWshTypesTbl.Cells(i, _
                    g_yWSH_TYPE_TBL_DEFINED_COL) Then
                WshTypeNm = Trim(rngdoWshTypesTbl.Cells(i, _
                        g_yWSH_TYPE_TBL_NM_COL))
                i = yWshTypesInTable '...break loop
            End If
        Next i
    End If
End If

'Set the caption for the form...
sFormCaption = sWSH_DIMEN_FORM_TTL & WshTypeNm
If g_bWizardMode Then sFormCaption = _
        sSTEP_WshTypeNm_CAP & sFormCaption
'If the form for this WshType doesn't exist create it...
If Not Me.WshTypeDimensFormsColl.ContainsItem(WshTypeNm) Then
    Me.WshTypeDimensFormsColl.Add _
            g_cObjFactory.CreateWshTypeDimensForm( _
                    WsType:=WshTypeNm, _
                    DictCompMeth:=TextCompare, _
                    FormCaption:=sFormCaption, _
                    RanOkRESULT:=bCalledProcedureRanOk)
               If Not bCalledProcedureRanOk Then _
                        Err.Raise g_lERR_HANDLED
End If
'Populate the form...
If g_bStepThruMode Then Stop
g_cDimHandler.PopulateWshTypesDimenFormFmSrcWbkDimenTbl _
        WshTypeNm:=WshTypeNm, _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

'Set the WTD combo box...
If OpenedFmAnotherWTDForm Then
    Set frmWTD = Me.WshTypeDimensFormsColl.Item(WshTypeNm)
    frmWTD.IgnoreChngInWshTypeCmbBox = True
    For i = 1 To frmWTD.cmbWshType.ListCount
        If StrComp(WshTypeNm, frmWTD.cmbWshType.List(i - 1, 0), _
                vbTextCompare) = 0 Then
            frmWTD.cmbWshType.ListIndex = i - 1
            i = frmWTD.cmbWshType.ListCount
        End If
    Next i
    frmWTD.IgnoreChngInWshTypeCmbBox = False
End If

Me.WshTypeDimensFormsColl.Item(WshTypeNm).Show
If g_bStepThruMode Then Stop

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
```

```
FormsHandler - 26

        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Public Sub OpenAndPopulateDataWshsForm(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "OpenAndPopulateDataWshsForm()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Const yWSHS_WE_CREATE_IN_DATA_WBK As Byte = 2
Const sSTEP_TTL As String = "STEP 3 OF 8:  "
Const sFILE_DIMEN_FORM_TTL As String = _
    "Select Data Worksheets"
Dim sCaption As String
Dim asWshNames() As String
Dim ySrcWbkPossibleDataWshs As Byte

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
sCaption = sFILE_DIMEN_FORM_TTL
If g_bWizardMode Then sCaption = sSTEP_TTL & sCaption
Set m_frmImportWshs = _
    g_cObjFactory.CreateWshsToImportForm( _
            FormCaption:=sCaption, _
            RanOkRESULT:=bCalledProcedureRanOk)
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
ySrcWbkPossibleDataWshs = _
    g_wbSrcData.Worksheets.Count - yWSHS_WE_CREATE_IN_DATA_WBK
ReDim asWshNames(0 To ySrcWbkPossibleDataWshs - 1)
'If g_bStepThruMode Then Stop
asWshNames() = CreateArraySourcWbkWshNames( _
        NmbrPossibleDataWorksheets:=ySrcWbkPossibleDataWshs, _
        RanOkRESULT:=bCalledProcedureRanOk)
     If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
'Per Chirag Mehta, users will prefer to see worksheets in their original order...
'''Utilities.BubbleSortArray _
''        arr:=asWshNames, _
''        RanOkRESULT:=bCalledProcedureRanOk
'''    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
'Populate list box...
m_frmImportWshs.lstWshsToImport.List = asWshNames
'Identify previously selected Worksheets-To-Import...
SelectWshsToImportFromSrcWbkValList _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
m_frmImportWshs.Show

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
```

```
FormsHandler - 27

        End If
End Sub
'================================================================================
Public Sub SetFileDimenDataFmFormMaestro(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "SetFileDimenDataFmFormMaestro()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Const yPAIRED_CTRLS As Byte = 12
Dim dimenFile As Dimension
Dim objTxt As MSForms.control
Dim txtCtrl As MSForms.TextBox
Dim refCtrl As RefEdit.RefEdit
Dim enumProblemScope As ProjDimenScope
Dim enumScale As ProjAmtScale
Dim sProblemField As String
Dim sProblemRowDimen As String
Dim sProblemColDimen As String
Dim sScale As String
Dim iScaleListIndex As Integer
Dim yUndefinedDimens As Byte
Dim i As Byte
Dim bInvalidRangeAddr As Boolean
Dim bMultiCellRangeAddr As Boolean
Dim bMultiRowColRangeAddr As Boolean
Dim bDupedDimenDefinitions As Boolean
Dim bUserAddedDimens As Boolean
Dim bUserRemovedDimens As Boolean
Dim bAllMandDimensDefined As Boolean
Dim bScaleSpecified As Boolean

'Dim bUndefinedMandDimens As Boolean

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'If g_bStepThruMode Then Stop
'Trap for bad range references...
bInvalidRangeAddr = AnyInvalidRangeReferences( _
    DimenScope:=projScopeFile, _
    InvalidFieldRESULT:=sProblemField)
'Notify user if there's a problem...
If bInvalidRangeAddr Then
    g_cMsgBoxHandler.SetInvalidRangeRefMsg sProblemField
    g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbExclamation
    GoTo ExitPoint
End If

'If g_bStepThruMode Then Stop
'Trap for multi-cell range references...
bMultiCellRangeAddr = AnyMultiCellRangeReferences( _
        DimenScope:=projScopeFile, _
        InvalidFieldRESULT:=sProblemField, _
        RanOkRESULT:=bCalledProcedureRanOk)
If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
'Notify user if there's a problem...
If bMultiCellRangeAddr Then
    g_cMsgBoxHandler.SetMultiCellRangeRefMsg sProblemField
    g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbExclamation
    GoTo ExitPoint
End If

'If g_bStepThruMode Then Stop
'Grab scaling selection.
'First see if user has even made a selection...
iScaleListIndex = m_frmFileMandDimens.cmbScale.ListIndex
If iScaleListIndex = -1 Then
    enumScale = projScaleNoSelection
Else
```

```
    sScale = _
        Trim(m_frmFileMandDimens.cmbScale.List(iScaleListIndex, 0))
End If

'If user HAS made a selection identify it...
If iScaleListIndex > -1 Then
    Select Case sScale
        Case g_sSCALE_SEE_WSH
            enumScale = projScaleSeeWshType
        Case g_sSCALE_NONE
            enumScale = projScaleNone
        Case g_sSCALE_THOUS
            enumScale = projScaleThous
        Case g_sSCALE_MILLS
            enumScale = projScaleMills
        Case g_sSCALE_BILLS
            enumScale = projScaleBills
    End Select
End If

'Set our boolean, for use at end of procedure...
If enumScale <> projScaleNoSelection And enumScale <> projScaleSeeWshType Then _
    bScaleSpecified = True

If g_bStepThruMode Then Stop
'Trap for user adding or removing a dimension definitions at the file level...
If m_bFileDimensPreviouslySet Then
    'Check for a user adding a new dimension defintion...
    Dim saNewDimens() As String
    Dim ansProceed As VbMsgBoxResult
    Dim yNewDimens As Byte
    'Check pseudo-dimensions first...
    If Not m_frmFileMandDimens.DimensPopulatedOnOpen.ContainsItem( _
            ItemName:=g_sDIMEN_SCALE, _
            CompMeth:=vbTextCompare) Then
        If enumScale > projScaleSeeWshType Then
            ReDim Preserve saNewDimens(0 To yNewDimens)
            saNewDimens(yNewDimens) = g_sDIMEN_SCALE
            yNewDimens = yNewDimens + 1
        End If
    End If
    '*** CHECK HERE FOR AMT TYPE IF EVER USED....
    'Now check on addition of real demensions...
    For Each objTxt In m_frmFileMandDimens.TextBoxesByCtrlNm
        If StrComp(Trim(objTxt.value), vbNullString) <> 0 Then
            If Not m_frmFileMandDimens.DimensPopulatedOnOpen.ContainsItem( _
                    ItemName:=objTxt.Tag, _
                    CompMeth:=vbTextCompare) Then
                ReDim Preserve saNewDimens(0 To yNewDimens)
                saNewDimens(yNewDimens) = objTxt.Tag
                yNewDimens = yNewDimens + 1
            End If
        End If
    Next objTxt
    For Each refCtrl In m_frmFileMandDimens.RefEditsByCtrlNm
        If StrComp(refCtrl.value, vbNullString) <> 0 Then
            If Not m_frmFileMandDimens.DimensPopulatedOnOpen.ContainsItem( _
                    ItemName:=refCtrl.Tag, _
                    CompMeth:=vbTextCompare) Then
                ReDim Preserve saNewDimens(0 To yNewDimens)
                saNewDimens(yNewDimens) = refCtrl.Tag
                yNewDimens = yNewDimens + 1
            End If
        End If
    Next refCtrl
    'Notify user if there are any new file dimensions...
    If g_bStepThruMode Then Stop
    If yNewDimens > 0 Then
        bUserAddedDimens = True
        g_cMsgBoxHandler.AskAddNewFileDimensionsQuest saNewDimens
        ansProceed = g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbQuestion + vbYesNoCancel)
```

```
        If ansProceed = vbYes Then
            For i = 1 To yNewDimens
                g_cDimHandler.RemoveDimenDefinitionFmAllWshTypes _
                        DimenNm:=saNewDimens(i - 1), _
                        RanOkRESULT:=bCalledProcedureRanOk
                    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
            Next i
        Else
            GoTo ExitPoint
        End If
    End If
    '************************
    'Check for a user REMOVING a dimension definition...
    '************************
    If g_bStepThruMode Then Stop
    Dim objDimen As Dimension
    Dim sDimenNm As String
    Dim saDeletedDimens() As String
    Dim yDeletedDimens As Byte
    Dim bDimenStillDefined As Boolean
    For Each objDimen In m_frmFileMandDimens.DimensPopulatedOnOpen
        bDimenStillDefined = False '...reset
        'Handle our pseudo-dimensions first...
        sDimenNm = Trim(objDimen.name)
        If StrComp(sDimenNm, g_sDIMEN_SCALE, vbTextCompare) = 0 Then
            'Handle no scale selection made now...
            If m_frmFileMandDimens.cmbScale.ListIndex = -1 Then
                saDeletedDimens(0) = g_sDIMEN_SCALE
                yDeletedDimens = 1
            Else '... there is still a selection made
                Dim yScaleIndex As Byte
                Dim sChosenScale As String
                yScaleIndex = m_frmFileMandDimens.cmbScale.ListIndex
                sChosenScale = _
                        m_frmFileMandDimens.cmbScale.List(yScaleIndex, 0)
                'Treat "Defined at Wsh Level" the same as removing a scale _
                definition at the file level...
                If StrComp(Trim(sChosenScale), _
                        g_sSCALE_SEE_WSH, vbTextCompare) = 0 Then
                    saDeletedDimens(0) = g_sDIMEN_SCALE
                    yDeletedDimens = 1
                End If
            End If
        Else '...handle all of our regular dimensions
            'Check appropriate text box...
            Set txtCtrl = m_frmFileMandDimens.TextBoxesByDimenNm.Item(sDimenNm)
            If StrComp(Trim(txtCtrl.value), vbNullString) <> 0 Then _
                bDimenStillDefined = True
            'If so needed check appropriate refEdit box...
            If Not bDimenStillDefined Then
                Set refCtrl = m_frmFileMandDimens.RefEditsByDimenNm.Item(sDimenNm)
                If StrComp(refCtrl.value, vbNullString) <> 0 Then _
                    bDimenStillDefined = True
            End If
            'Handle a dimension which is no longer defined at the file level...
            If Not bDimenStillDefined Then
                ReDim Preserve saDeletedDimens(0 To yDeletedDimens) As String
                saDeletedDimens(yDeletedDimens) = sDimenNm
                yDeletedDimens = yDeletedDimens + 1
            End If
        End If
    Next objDimen
    'Deal with any removed dimensions...
    If yDeletedDimens > 0 Then
        bUserRemovedDimens = True
        g_cMsgBoxHandler.AskRemoveFileDimensionsQuest saDeletedDimens
        ansProceed = _
                g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbQuestion + _
                vbYesNoCancel)
        If ansProceed = vbYes Then
            'Set Mandatory Dimensions Defined value to false for _
```

```
            all worksheet types...
            Dim wsValLists As Worksheet
            Dim rngWshTypeStatusTbl As Range
            Dim c As Range
            Dim yWshTypes As Byte
            If g_bStepThruMode Then Stop
            Set wsValLists = _
                    g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
            Set rngWshTypeStatusTbl = wsValLists.Range("valtbldodynWshTypeInfo")
            yWshTypes = rngWshTypeStatusTbl.Rows.Count
            For i = 1 To yWshTypes
                rngWshTypeStatusTbl.Cells(i, _
                        g_yWSH_TYPE_TBL_DEFINED_COL).value = False
            Next i
        Else
            GoTo ExitPoint
        End If
    End If
End If

If g_bStepThruMode Then Stop
'Trap for user specifying too many dimensions!  (There must be at least one _
column and at least one row dimension specified at WshType level.)...

'Start by finding the number of underfined dimensions...
For Each objTxt In m_frmFileMandDimens.TextBoxesByDimenNm
    If StrComp(Trim(objTxt.Tag), g_sDIMEN_SCALE, vbTextCompare) <> 0 Then
        Set refCtrl = _
                m_frmFileMandDimens.RefEditsByDimenNm.Item(Trim(objTxt.Tag))
        Set txtCtrl = objTxt '...so we can get text/value property
        If StrComp(Trim(txtCtrl.value), vbNullString) = 0 And _
                StrComp(Trim(refCtrl.value), vbNullString) = 0 Then
            yUndefinedDimens = yUndefinedDimens + 1
        End If
    End If
Next objTxt
If yUndefinedDimens < 2 Then
    g_cMsgBoxHandler.SetTooManyFileDimensMsg
    g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbInformation
    GoTo ExitPoint
End If


'******************
'If here then we have valid entries.
'******************
If g_bStepThruMode Then Stop

ClearAndRepopulateDimenTblFmWbkDimenForm _
        AmtScale:=enumScale, _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
If Not g_bWizardMode Then ThisWorkbook.Activate
m_bFileDimensPreviouslySet = True '...set module-level flag
'If NextStage enum was set only to FileDimens then we can bump it up...
If Not m_bDataWshsPreviouslySet Then _
        RibbonHandler.SetNextStage projNxtStgIDDataWshs

If g_bStepThruMode Then Stop
'Update our File Dimensions collection...
g_cDimHandler.ClearFileDimensCollection RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
For Each objTxt In m_frmFileMandDimens.TextBoxesByDimenNm
    Set txtCtrl = objTxt
    If StrComp(Trim(txtCtrl.value), vbNullString) <> 0 Then
        Set dimenFile = g_cDimHandler.AllDimensions.Item(objTxt.Tag)
        g_cDimHandler.FileDimensions.Add dimenFile
    End If
Next objTxt
For Each refCtrl In m_frmFileMandDimens.RefEditsByDimenNm
    If StrComp(Trim(refCtrl.value), vbNullString) <> 0 Then
```

```
        Set dimenFile = g_cDimHandler.AllDimensions.Item(refCtrl.Tag)
        g_cDimHandler.FileDimensions.Add dimenFile
    End If
Next refCtrl

If g_bStepThruMode Then Stop
'If the user removed file dimensions then we have already set the _
All-Mandatory-Dimensions-Defined flag in the XL WshType status table to _
FALSE.  However, if the user added dimension AND DID NOT REMOVE ANY there is _
the possibility that some of those flags should now be set to TRUE.  Ergo...
If bUserAddedDimens And Not bUserRemovedDimens Then _
    g_cDimHandler.AsNeededUpdateMandDimenDefinedFlagForWshTypes _
            ScaleSetAtFileLevel:=bScaleSpecified, _
            RanOkRESULT:=bCalledProcedureRanOk
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

g_wbSrcData.Save

'Next steps...
m_frmFileMandDimens.Hide
If g_bWizardMode Then
    RibbonHandler.ResetRibbon
    If g_bStepThruMode Then Stop
    OpenAndPopulateDataWshsForm _
            RanOkRESULT:=bCalledProcedureRanOk
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
Else
    If g_bUserRenamedOrDeletedDataWshs Then
        RibbonHandler.SetNextStageAndResetRibbon projNxtStgIDDataWshs
    Else
        If g_cDimHandler.AnyWshTypesWithUndefinedMandDimens( _
                RanOkRESULT:=bCalledProcedureRanOk) Then
            RibbonHandler.ResetRibbon
        Else
            RibbonHandler.SetNextStageAndResetRibbon projNxtStgImportData
        End If
    End If
End If


ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Public Sub SetWshTypeDimenDataFmFormMaestro( _
    ByRef RanOkRESULT As Boolean, _
    ByVal WshTypeNm As String)
'Error-handling declarations...
Const sSOURCE As String = "SetWshTypeDimenDataFmFormMaestro()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim Dimen As Dimension
```

```
Dim collUnidentDimens As Dimensions
Dim lblFileDimens As MSForms.label
Dim refWsh As RefEdit.RefEdit
Dim refRow As RefEdit.RefEdit
Dim refCol As RefEdit.RefEdit
Dim enumProblemScope As ProjDimenScope
Dim enumScale As ProjAmtScale
Dim sProblemField As String
Dim sProblemRowDimen As String
Dim sProblemColDimen As String
Dim sScale As String
Dim iScaleListIndex As Integer
Dim i As Byte
Dim bInvalidRangeAddr As Boolean
Dim bMultiCellRangeAddr As Boolean
Dim bMultiRowColRangeAddr As Boolean
Dim bDupedDimenDefinitions As Boolean
Dim bAllMandDimensDefined As Boolean
Dim bScaleSpecified As Boolean

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True
WshTypeNm = Trim(WshTypeNm)

'If g_bStepThruMode Then Stop
'Trap for bad range references...
bInvalidRangeAddr = AnyInvalidRangeReferences( _
    DimenScope:=projScopeWshType, _
    InvalidFieldRESULT:=sProblemField, _
    WshTypeNm:=WshTypeNm)
'Notify user if there's a problem...
If bInvalidRangeAddr Then
    g_cMsgBoxHandler.SetInvalidRangeRefMsg sProblemField
    g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbExclamation
    GoTo ExitPoint
End If

'If g_bStepThruMode Then Stop
'Trap for user having specified a dimension in > 1 location...
bDupedDimenDefinitions = AnyDupeDimensionDefinitions( _
        WshTypeNm:=WshTypeNm, _
        DupedDimenDefRESULT:=sProblemField, _
        RanOkRESULT:=bCalledProcedureRanOk)
'Notify user if there's a problem...
If bDupedDimenDefinitions Then
    g_cMsgBoxHandler.SetDupeDimensionDefinitionsMsg sProblemField
    g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbExclamation
    GoTo ExitPoint
End If

'If g_bStepThruMode Then Stop
'Trap for multi-cell range references...
bMultiCellRangeAddr = AnyMultiCellRangeReferences( _
        DimenScope:=projScopeWshType, _
        InvalidFieldRESULT:=sProblemField, _
        RanOkRESULT:=bCalledProcedureRanOk, _
        WshTypeNm:=WshTypeNm)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
'Notify user if there's a problem...
If bMultiCellRangeAddr Then
    g_cMsgBoxHandler.SetMultiCellRangeRefMsg sProblemField
    g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbExclamation
    GoTo ExitPoint
End If

'If g_bStepThruMode Then Stop
'Trap for multi-row row references and multi-col column references...
bMultiRowColRangeAddr = AnyInvalidRowColRangeReferences( _
        DimScopeRESULT:=enumProblemScope, _
        InvalidRowDimenRESULT:=sProblemRowDimen, _
```

```
        invalidColDimenRESULT:=sProblemColDimen, _
        RanOkRESULT:=bCalledProcedureRanOk, _
        WshTypeNm:=WshTypeNm)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
'Notify user if there's a problem...
If bMultiRowColRangeAddr Then
    g_cMsgBoxHandler.SetMultiRowColRangeRefMsg _
            DimScope:=enumProblemScope, _
            ProblemField:=sProblemRowDimen
    g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbExclamation
    GoTo ExitPoint
End If

If g_bStepThruMode Then Stop
'Grab scaling selection.
'First see if user has even made a selection...
Dim frmWTD As frmWshTypeDimens
Set frmWTD = Me.WshTypeDimensFormsColl.Item(WshTypeNm)
iScaleListIndex = frmWTD.cmbScale.ListIndex
If iScaleListIndex = -1 Then
    enumScale = projScaleNoSelection
Else
    sScale = _
        Trim(frmWTD.cmbScale.List(iScaleListIndex, 0))
End If

'If user HAS made a selection identify it...
If iScaleListIndex > -1 Then
    Select Case sScale
        Case g_sSCALE_SEE_WSH
            enumScale = projScaleSeeWshType
        Case g_sSCALE_NONE
            enumScale = projScaleNone
        Case g_sSCALE_THOUS
            enumScale = projScaleThous
        Case g_sSCALE_MILLS
            enumScale = projScaleMills
        Case g_sSCALE_BILLS
            enumScale = projScaleBills
    End Select
End If

If g_bStepThruMode Then Stop
'Trap for user not having identified all mandatory dimensions...
Set collUnidentDimens = New Dimensions
For Each Dimen In g_cDimHandler.AllDimensions
    If Dimen.RequirementType = projDimenMandatory Then
        'Find the label corressponding to this dimension...
        Set lblFileDimens = _
                frmWTD.FileDimenLblsByDimenNm.Item(Dimen.name)
        If StrComp(Trim(lblFileDimens.Caption), vbNullString) = 0 Then
            'Find the Wsh, Row, & Col RefEdits corresponding to _
            this dimension...
            Set refWsh = _
                    frmWTD.WshDimenRefEditsByDimenNm.Item(Dimen.name)
            Set refRow = _
                    frmWTD.RowDimenRefEditsByDimenNm.Item(Dimen.name)
            Set refCol = _
                    frmWTD.ColDimenRefEditsByDimenNm.Item(Dimen.name)
            'See if any 1 of the RefEdit dimensions has a value _
            specified.  If not, add the dimension to our collection of _
            unidentified dimensions...
            If StrComp(Trim(refWsh.value), vbNullString) = 0 And _
                    StrComp(Trim(refRow.value), vbNullString) = 0 And _
                    StrComp(Trim(refCol.value), vbNullString) = 0 Then
                collUnidentDimens.Add Dimen
            End If
        End If
    End If
Next Dimen
'Now check on scale (NOTE:  at WshType level we do not offer the _
```

```
"Define at Wsh Level" option)...
If StrComp(Trim(frmWTD.lblFileDimScale.Caption), _
        vbNullString) <> 0 Then bScaleSpecified = True
If frmWTD.cmbScale.ListIndex > -1 Then bScaleSpecified = True
'Notify user if we have unspecified manadatory dimensions (or scaling)...
If collUnidentDimens.Count > 0 Or Not bScaleSpecified Then
    Dim sUndefDimensTbl As String
    Dim ansCloseFormAnyway As VbMsgBoxResult
    For i = 1 To collUnidentDimens.Count
        sUndefDimensTbl = _
                sUndefDimensTbl & vbTab & _
                collUnidentDimens(i).name & vbCr
    Next i
    'Add scale, if needed...
    If Not bScaleSpecified Then sUndefDimensTbl = _
            sUndefDimensTbl & vbTab & g_sDIMEN_SCALE
    g_cMsgBoxHandler.AskCloseEvenThoUndefinedMandDimensQuest _
            sUndefDimensTbl
    ansCloseFormAnyway = _
            g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbYesNoCancel)
    If ansCloseFormAnyway = vbNo Or ansCloseFormAnyway = vbCancel Then
        GoTo ExitPoint
    End If
Else
    bAllMandDimensDefined = True
End If

'******************
'If here then we have valid entries.
'******************
If g_bStepThruMode Then Stop
PopulateDimenTblFmWshTypeDimenForm _
        WshTypeNm:=WshTypeNm, _
        AmtScale:=enumScale, _
        UndefinedMandDimens:=bAllMandDimensDefined, _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

'Next steps...
UserOKClosesWshDimenFormMaestro _
        WshTypeName:=WshTypeNm, _
        AllMandDimensDefined:=bAllMandDimensDefined, _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED


ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'=============================================================================
Public Sub UserOKClosesWshsToImportForm(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
```

```
Const sSOURCE As String = "UserOKClosesWshsToImportForm()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim collDeselectedWshs As Collection
Dim collEmptyWshTypes As Collection
Dim saWshsToImport() As String
Dim saDeselectedWshs() As String
Dim iWshsToImport As Integer
Dim i As Integer
Dim bUserSelectedAWorksheet As Boolean
Dim bUserAddedNewWsh As Boolean
Dim bUserCxlsAtAddWshsPrompt As Boolean
Dim bUserDeselectedWshs As Boolean
Dim bWshTypeEmpty As Boolean
Dim bUserSaysDeleteEmptyWshTypes As Boolean
Dim bUserAddedNewWshsToSingleType As Boolean
Dim bMustRecreateALLType As Boolean

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'If g_bStepThruMode Then Stop
'Trap for user not having selected at least one data worksheet...
For i = 0 To m_frmImportWshs.lstWshsToImport.ListCount - 1
    If m_frmImportWshs.lstWshsToImport.Selected(i) Then
        bUserSelectedAWorksheet = True
        i = m_frmImportWshs.lstWshsToImport.ListCount - 1 '...break loop
    End If
Next i
If Not bUserSelectedAWorksheet Then
    g_cMsgBoxHandler.SetNoDataWshsSelectedMsg
    g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbExclamation
    GoTo ExitPoint
End If

'If g_bStepThruMode Then Stop
'Build an array of the worksheets to import...
For i = 0 To m_frmImportWshs.lstWshsToImport.ListCount - 1
    If m_frmImportWshs.lstWshsToImport.Selected(i) Then
        ReDim Preserve saWshsToImport(0 To iWshsToImport)
        saWshsToImport(iWshsToImport) = _
                m_frmImportWshs.lstWshsToImport.List(i, 0)
        iWshsToImport = iWshsToImport + 1
    End If
Next i

If g_bStepThruMode Then Stop
'Handle user adding new worksheets to import...
If m_bDataWshsPreviouslySet Then
    CheckForAndHandleUserAddingDataWshs _
            DataWshsInFormRESULT:=saWshsToImport, _
            UserAddedWshsRESULT:=bUserAddedNewWsh, _
            UserCxlsAddingWshsToSingleWshTypeRESULT:=bUserCxlsAtAddWshsPrompt, _
            UserAddedNewWshsToSingleTypeRESULT:=bUserAddedNewWshsToSingleType, _
            RanOkRESULT:=bCalledProcedureRanOk
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    If bUserCxlsAtAddWshsPrompt Then GoTo ExitPoint

    If g_bStepThruMode Then Stop
    'Handle user de-selecting worksheets to import...
    Set collDeselectedWshs = New Collection
    Set collEmptyWshTypes = New Collection
    CheckForAndHandleUserRemovingDataWshs _
            SelectedDataWshsInForm:=saWshsToImport, _
            DeselectedWshs:=collDeselectedWshs, _
            ToBeEmptyWshTypes:=collEmptyWshTypes, _
            UserSaysDeleteWshTypesRESULT:=bUserSaysDeleteEmptyWshTypes, _
            EveryWshTypeWillBeEmptyRESULT:=bMustRecreateALLType, _
            RanOkRESULT:=bCalledProcedureRanOk
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
```

```
    If collDeselectedWshs.Count > 0 Then bUserDeselectedWshs = True
    If collEmptyWshTypes.Count > 0 Then bWshTypeEmpty = True
End If

If g_bStepThruMode Then Stop
If bUserDeselectedWshs Then
    If bWshTypeEmpty Then
        If bUserSaysDeleteEmptyWshTypes Then
            If bMustRecreateALLType Then
                AddALLWshType RanOkRESULT:=bCalledProcedureRanOk
                    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
            End If
            Dim sEmptyWshType As String
            For i = 1 To collEmptyWshTypes.Count
                sEmptyWshType = collEmptyWshTypes.Item(i)
                RemoveWshTypeFmXLValListsAndDimenLocTbl _
                        WshTypeNm:=sEmptyWshType, _
                        RanOkRESULT:=bCalledProcedureRanOk
                    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
                'Remove the worksheet type from the WshType collection...
                If g_cDimHandler.WorksheetTypes.ContainsItem(sEmptyWshType) Then
                    g_cDimHandler.WorksheetTypes.Remove _
                            index:=sEmptyWshType, _
                            RanOkRESULT:=bCalledProcedureRanOk
                        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
                End If
            Next i
        Else
            'Handle as if there are no empty WshTypes - i.e., do nothing here...
        End If
    Else '...no worksheet TYPES are left empty
        For i = 1 To collDeselectedWshs.Count
            RemoveDataWshFmWshNmWshTypeMapTbl _
                    ExDataWsh:=collDeselectedWshs.Item(i), _
                    RanOkRESULT:=bCalledProcedureRanOk
                If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        Next i
    End If
End If

'If here then the user picked at least one wsh to import...
PostDataWshsFmFormToSrcWbkValList _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

If g_bWizardMode Or _
        bUserAddedNewWsh Or _
        bUserDeselectedWshs Or _
        Not m_bDataWshsPreviouslySet Then
    g_wbSrcData.Save
End If

m_frmImportWshs.Hide
m_bDataWshsPreviouslySet = True

'If g_bStepThruMode Then Stop
If g_bWizardMode Then
    'Set ribbon...
    RibbonHandler.SetNextStageAndResetRibbon projNxtStgCreateWshTyps
    If g_bStepThruMode Then Stop
    g_cFormsHandler.OpenAndPopulateAddRemWshTypeForm _
            RanOkRESULT:=bCalledProcedureRanOk
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
Else '...previously opened workbook
    If g_bUserRenamedOrDeletedDataWshs Then
        RibbonHandler.SetNextStage projNxtStgAssocWshTyps
    Else
        If bUserAddedNewWsh Then
            If bUserAddedNewWshsToSingleType Then
                'Do nothing - drop down to checking whether we have wsh _
                types with undefined mandatory dimensions...
```

```
            Else
                'Our default is to disable the AssociateWshTypes button _
                when there is only one WshType.  However, in the case where _
                the user opted not to associate newly-specified data worksheets _
                with his one worksheet type we need to allow him to use the _
                Associate button...
                RibbonHandler.SetEnableAssocWshTypesSpecCaseFlag True
                RibbonHandler.SetNextStageAndResetRibbon projNxtStgAssocWshTyps
                GoTo ExitPoint
            End If
        End If
        If bUserDeselectedWshs Then
            If bWshTypeEmpty Then
                If bUserSaysDeleteEmptyWshTypes Then
                    'Do nothing - drop down to checking whether we have wsh _
                    types with undefined mandatory dimensions...
                Else
                    RibbonHandler.SetNextStageAndResetRibbon _
                            projNxtStgAssocWshTyps
                    GoTo ExitPoint
                End If
            Else
                'Do nothing - drop down to checking whether we have wsh _
                types with undefined mandatory dimensions...
            End If
        End If
        If g_cDimHandler.AnyWshTypesWithUndefinedMandDimens( _
                RanOkRESULT:=bCalledProcedureRanOk) Then
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
            RibbonHandler.SetNextStage projNxtStgWshTypeDimens
        Else
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
            RibbonHandler.SetNextStage projNxtStgImportData
        End If
        'Now see if we have to ratchet things back a bit...
        If Not m_bWshTypeWshNmPreviouslySet Then RibbonHandler.SetNextStage projNxtStgAssocWshTyps
        If Not m_bWshTypesPreviouslySet Then RibbonHandler.SetNextStage projNxtStgCreateWshTyps
    End If
    RibbonHandler.ResetRibbon
End If

ExitPoint:
On Error Resume Next
'''Must-run procedure/function clean-up code...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
            If g_bDebugMode Then Stop
            'So we can step through the code which caused the error...
            Resume
        Else
            Resume ExitPoint
        End If
End Sub
'================================================================================
Public Sub PostDataWshsFmFormToSrcWbkValList(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "PostDataWshsFmFormToSrcWbkValList()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wsSrcWbkValLists As Worksheet
Dim i As Byte
```

```
FormsHandler - 38

Dim j As Byte

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
'Handle selected worksheets...
Set wsSrcWbkValLists = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
wsSrcWbkValLists.Range("vallstdodynWshsToImport").ClearContents

j = 1
For i = 0 To m_frmImportWshs.lstWshsToImport.ListCount - 1
    If m_frmImportWshs.lstWshsToImport.Selected(i) Then
        wsSrcWbkValLists.Range("vallstdodynWshsToImport").Cells(j, 1) = _
            m_frmImportWshs.lstWshsToImport.List(i, 0)
        j = j + 1
    End If
Next i

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Public Sub OpenAndPopulateAddRemWshTypeForm(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "OpenAndPopulateAddRemWshTypeForm()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Const sSTEP_TTL As String = "STEP 4 OF 8:  "
Const sADD_RMV_WSH_TYPE_TTL As String = "Add/Remove Worksheet Types"
Dim wsSrcWbkValList As Worksheet
Dim rngdoWshTypeTbl As Range
Dim saWshTypes() As String
Dim sFormCaption As String
Dim yWshTypesOnSrcWbk As Byte
Dim i As Byte

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
'Set up form...
Set m_frmAddRemWshTypes = New frmAddRemWshTypes
sFormCaption = sADD_RMV_WSH_TYPE_TTL
If g_bWizardMode Then sFormCaption = sSTEP_TTL & sFormCaption
m_frmAddRemWshTypes.Caption = sFormCaption

Set wsSrcWbkValList = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
Set rngdoWshTypeTbl = wsSrcWbkValList.Range("valtbldodynWshTypeInfo")
yWshTypesOnSrcWbk = rngdoWshTypeTbl.Rows.Count
```

```
'This should NEVER happen, but just in case...
If yWshTypesOnSrcWbk = 0 Then
    rngdoWshTypeTbl.Cells(1, g_yWSH_TYPE_TBL_NM_COL) = g_sWSH_TYPE_ALL_PC
    rngdoWshTypeTbl.Cells(1, g_yWSH_TYPE_TBL_DEFINED_COL).value = False
    yWshTypesOnSrcWbk = 1
End If

'Populate array with worksheet types listed in source wbk...
ReDim saWshTypes(0 To yWshTypesOnSrcWbk - 1)
For i = 0 To yWshTypesOnSrcWbk - 1
    saWshTypes(i) = rngdoWshTypeTbl.Cells(i + 1, g_yWSH_TYPE_TBL_NM_COL)
Next i

'Sort the array, then pop it into the form...
Utilities.BubbleSortArray arr:=saWshTypes, RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
For i = 0 To yWshTypesOnSrcWbk - 1
    m_frmAddRemWshTypes.lstTypes.AddItem saWshTypes(i)
    If StrComp(Trim(saWshTypes(i)), g_sWSH_TYPE_ALL_PC, vbTextCompare) = 0 Then _
        m_frmAddRemWshTypes.ALLTypeSpecified = True
Next i

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
m_frmAddRemWshTypes.Show
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Public Sub UserAddsWshType(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "UserAddsWshType()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim ansProceed As VbMsgBoxResult
Dim saWshTypes() As String
Dim yExistingTypes As Byte
Dim i As Byte
Dim bInclDimDefLossReminder As Boolean
Static bToldUserAboutNeedToDefineDimensForNewWshType As Boolean

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
If g_bStepThruMode Then Stop
'This should NEVER be 0...
yExistingTypes = m_frmAddRemWshTypes.lstTypes.ListCount

bInclDimDefLossReminder = g_cDimHandler.AnyWshTypeDimensDefined( _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
```

```
'Trap for user adding the ALL type when there are items in the list box...
If StrComp(Trim(m_frmAddRemWshTypes.txtType), _
        g_sWSH_TYPE_ALL_PC, vbTextCompare) = 0 And yExistingTypes > 0 Then
    g_cMsgBoxHandler.AskWantToRemoveWshsTypesBcAddingAllTypeQuest _
            IncludeWillLoseDimenDefsReminder:=bInclDimDefLossReminder
    ansProceed = g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbYesNo)
    If ansProceed = vbYes Then
        m_frmAddRemWshTypes.lstTypes.Clear
        m_frmAddRemWshTypes.lstTypes.AddItem g_sWSH_TYPE_ALL_PC
        m_frmAddRemWshTypes.ALLTypeSpecified = True
    End If
    m_frmAddRemWshTypes.txtType = vbNullString
    GoTo ExitPoint
End If

'Trap for user adding a new wsh type when the list box contains the ALL type...
If m_frmAddRemWshTypes.ALLTypeSpecified And _
        StrComp(Trim(m_frmAddRemWshTypes.txtType), vbNullString) <> 0 Then
    g_cMsgBoxHandler.AskWantToRemoveTheAllWshTypeQuest _
            IncludeWillLoseDimenDefsReminder:=bInclDimDefLossReminder
    ansProceed = g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbYesNo)
    If ansProceed = vbYes Then
        m_frmAddRemWshTypes.lstTypes.Clear
        m_frmAddRemWshTypes.lstTypes.AddItem Trim(m_frmAddRemWshTypes.txtType)
        m_frmAddRemWshTypes.ALLTypeSpecified = False
    End If
    m_frmAddRemWshTypes.txtType = vbNullString
    GoTo ExitPoint
End If

yExistingTypes = m_frmAddRemWshTypes.lstTypes.ListCount '...reset

'If this is a normal case of adding a new type - having nothing to do with _
the ALL type - we have that new type in the text box with nothing added _
yet to the list box...
If StrComp(Trim(m_frmAddRemWshTypes.txtType), vbNullString) <> 0 Then
    If m_bWshTypeWshNmPreviouslySet Then
        'This is a static variable, so we only warn the user once...
        If Not bToldUserAboutNeedToDefineDimensForNewWshType Then
            g_cMsgBoxHandler.AskAddWshTypeEvenThoMustSpecifyDimensQuest
            ansProceed = _
                    g_cMsgBoxHandler.ShowMsgBoxReturnAnswer( _
                    vbQuestion + vbYesNoCancel)
            bToldUserAboutNeedToDefineDimensForNewWshType = True
            If ansProceed <> vbYes Then
                m_frmAddRemWshTypes.txtType = vbNullString
                GoTo ExitPoint
            End If
        End If
    End If
    m_frmAddRemWshTypes.lstTypes.AddItem Trim(m_frmAddRemWshTypes.txtType)
    m_frmAddRemWshTypes.txtType = vbNullString
    yExistingTypes = m_frmAddRemWshTypes.lstTypes.ListCount '...reset
End If

'Resort our list...
ReDim saWshTypes(0 To yExistingTypes - 1)
'Populate the array with existing types...
If yExistingTypes > 0 Then
    For i = 0 To yExistingTypes - 1
        saWshTypes(i) = m_frmAddRemWshTypes.lstTypes.List(i, 0)
    Next i
End If
'Sort and update the list box...
Utilities.BubbleSortArray saWshTypes, bCalledProcedureRanOk
m_frmAddRemWshTypes.lstTypes.Clear
For i = 0 To yExistingTypes - 1
    m_frmAddRemWshTypes.lstTypes.AddItem saWshTypes(i)
Next i
```

```
FormsHandler - 41

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Public Sub UserRemovesWshTypes(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "UserRemovesWshType()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim ansProceed As VbMsgBoxResult
Dim saWshTypes() As String
Dim yExistingTypes As Byte
Dim ySelectedForRemoval As Byte
Dim i As Byte
Dim bAnyWshTypeDimensDefined As Boolean
Static bUserWarnedAboutRemovingWshTypes As Boolean

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
bAnyWshTypeDimensDefined = g_cDimHandler.AnyWshTypeDimensDefined( _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

yExistingTypes = m_frmAddRemWshTypes.lstTypes.ListCount
ReDim saWshTypes(0 To yExistingTypes - 1)
For i = 0 To yExistingTypes - 1
    If Not m_frmAddRemWshTypes.lstTypes.Selected(i) Then
        'User wants to keep the item...
        saWshTypes(i) = m_frmAddRemWshTypes.lstTypes.List(i, 0)
    Else
        'Track the number of items user wants to remove...
        ySelectedForRemoval = ySelectedForRemoval + 1
    End If
Next i

If ySelectedForRemoval = 0 Then GoTo ExitPoint

If bAnyWshTypeDimensDefined And Not bUserWarnedAboutRemovingWshTypes Then
    'Warn user about removing a worksheet type...
    g_cMsgBoxHandler.AskConfirmWantToRemoveWshTypeQuest
    ansProceed = _
            g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbQuestion + vbYesNoCancel)
    'So we only ask once...
    bUserWarnedAboutRemovingWshTypes = True
    If ansProceed <> vbYes Then GoTo ExitPoint
End If

'Trap for user removing the ALL type...
If ySelectedForRemoval = 1 And m_frmAddRemWshTypes.ALLTypeSpecified Then _
```

```
FormsHandler - 42

    m_frmAddRemWshTypes.ALLTypeSpecified = False

'Clear and repopulate list box...
m_frmAddRemWshTypes.lstTypes.Clear
For i = 0 To yExistingTypes - 1
    If StrComp(saWshTypes(i), vbNullString) <> 0 Then _
        m_frmAddRemWshTypes.lstTypes.AddItem saWshTypes(i)
Next i

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Public Sub UserOKClosesAddRemWshTypesFormMaestro(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "UserOKClosesAddRemWshTypesFormMaestro()"
Dim bCalledProcedureRanOk As Boolean
'''Operating code declarations...
Dim bOnly1WshType As Boolean

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
On Error GoTo ErrorHandler
If g_bStepThruMode Then Stop

'Trap for user having a WshType still sitting in the New Worksheet Type box...
If StrComp(Trim(m_frmAddRemWshTypes.txtType.value), vbNullString) <> 0 Then
    Dim ansIncl As VbMsgBoxResult
    g_cMsgBoxHandler.AskWantToIncludeTypeInTxtBoxQuest _
            Trim(m_frmAddRemWshTypes.txtType.value)
    ansIncl = g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbQuestion + vbYesNoCancel)
    Select Case ansIncl
        Case vbYes
            'Essentially, we click the Add button for the user...
            UserAddsWshType RanOkRESULT:=bCalledProcedureRanOk
                If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        Case vbNo
            'Clear the box, then go on...
            m_frmAddRemWshTypes.txtType = vbNullString
        Case vbCancel
            GoTo ExitPoint
    End Select
End If

'Trap for user having NO wsh types on form...
If m_frmAddRemWshTypes.lstTypes.ListCount = 0 Then
    g_cMsgBoxHandler.SetNoWshTypesIdentifiedMsg
    g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbExclamation
    GoTo ExitPoint
```

```
End If

If m_frmAddRemWshTypes.lstTypes.ListCount = 1 Then bOnly1WshType = True

'See if the ALL type is specified...
If bOnly1WshType And _
        StrComp(Trim(m_frmAddRemWshTypes.lstTypes.List(0, 0)), _
        g_sWSH_TYPE_ALL_PC, vbTextCompare) = 0 Then _
        m_frmAddRemWshTypes.ALLTypeSpecified = True

'Populate the 2 SL worksheets in the source workbook...
PostWshTypesFromForm _
        UsingALLType:=m_frmAddRemWshTypes.ALLTypeSpecified, _
        OnlyOneWshType:=bOnly1WshType, _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

'If the worksheet types collection exists clean it out...
If g_cDimHandler.WorksheetTypes.Count > 0 Then
    Dim objWshType As WshType
    For Each objWshType In g_cDimHandler.WorksheetTypes
        objWshType.RemoveAll '...removes all worksheets from the type
    Next objWshType
    g_cDimHandler.WorksheetTypes.RemoveAll
End If

'Repopulate wsh types collection...
g_cDimHandler.CreateWshTypesFromFile RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

If g_bStepThruMode Then Stop
m_frmAddRemWshTypes.Hide

'Handle ribbon, other next steps...
If g_bStepThruMode Then Stop
RibbonHandler.SetOnly1WshTypeFlag bOnly1WshType
If g_bWizardMode Then
    m_bWshTypesPreviouslySet = True
    If Not bOnly1WshType Then
        RibbonHandler.SetNextStageAndResetRibbon projNxtStgAssocWshTyps
        If g_bStepThruMode Then Stop
        g_cFormsHandler.OpenAndPopulateAssocWshTypeWithWshForm _
                RanOkRESULT:=bCalledProcedureRanOk
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    Else
        m_bWshTypeWshNmPreviouslySet = True
        RibbonHandler.SetNextStageAndResetRibbon projNxtStgWshTypeDimens
        g_cFormsHandler.OpenWshTypeDimenIDForm _
                OpenedFmAnotherWTDForm:=False, _
                RanOkRESULT:=bCalledProcedureRanOk, _
                WshTypeNm:=Trim(m_frmAddRemWshTypes.lstTypes.List(0, 0))
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    End If
Else
    'Set ribbon based on how far we have previously made it through the _
    wizard.  Start with the most restrictive and work from there...
        'One "trick" - regardless of how far the user got previously in the _
        wizard if he renamed or deleted data worksheets since he last used _
        this here little Conversion Tool we need to force him to go back _
        through all steps.  Thus, just for the moment...
    If g_bUserRenamedOrDeletedDataWshs Then m_bWshTypesPreviouslySet = False

    'NOTE:  I originally wrote this IF/THEN for the _
    "Not m_bWshTypesPreviouslySet" condition.  I then realized that the _
    next steps for the "Not m_bWshTypeWshNmPreviouslySet" are the same...
    If Not m_bWshTypesPreviouslySet Or Not m_bWshTypeWshNmPreviouslySet Then
        If bOnly1WshType Then
            RibbonHandler.SetNextStage projNxtStgWshTypeDimens
            'At the very end of its code the PostWshTypesFromForm procedure _
            checks to see if there is only one WshType.  If that is the _
            case it calls a sub procedure which populates the WshNm/WshType _
```

```
            table with all data worksheets and maps each one to the single _
            WshType...
            m_bWshTypeWshNmPreviouslySet = True
        Else
            RibbonHandler.SetNextStage projNxtStgAssocWshTyps
        End If
        m_bWshTypesPreviouslySet = True '...NOW set this
    End If

    'Handle the case where the user has already gone all the way through the _
    Associate Worksheet Types portion of the wizard...
    If m_bWshTypeWshNmPreviouslySet Then
        Dim bWshTypesHaveUndefinedMandDimens As Boolean
        bWshTypesHaveUndefinedMandDimens = _
                g_cDimHandler.AnyWshTypesWithUndefinedMandDimens( _
                RanOkRESULT:=bCalledProcedureRanOk)
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        If bWshTypesHaveUndefinedMandDimens Then
            RibbonHandler.SetNextStage projNxtStgWshTypeDimens
        Else
            RibbonHandler.SetNextStage projNxtStgImportData
        End If
    End If
    RibbonHandler.ResetRibbon
    m_bWshTypesPreviouslySet = True '...just to be sure
    If g_bStepThruMode Then Stop
End If

ExitPoint:
On Error Resume Next
'''Must-run procedure/function clean-up code...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Public Sub UserWantsToCxlCloseForm( _
        ByVal FormName As String, _
        Optional WshTypeName As String)
'Error-handling declarations...
Const sSOURCE As String = "UserWantsToCxlCloseForm()"
'We have to treat this sub as an entry point because if the user says "Yes" _
to msgbox Prompt then the form which called this procedure will not exist to _
catch any thrown error...
Const bSUB_IS_ENTRY_PT As Boolean = True
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim ansCloseForm As VbMsgBoxResult

On Error GoTo ErrorHandler
'Assume success until the procedure fails...

If g_bStepThruMode Then Stop
'CODE HERE...
Select Case FormName
    Case "frmWbkMandDimens"
        g_cMsgBoxHandler.AskReallyWantToCxlWorkOnFormQues
```

```
        ansCloseForm = g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbYesNo)
        If ansCloseForm = vbNo Then GoTo ExitPoint
        Unload m_frmFileMandDimens
        Set m_frmFileMandDimens = Nothing
    Case "frmWshsToImport"
        'If g_bStepThruMode Then Stop
        Unload m_frmImportWshs
        Set m_frmImportWshs = Nothing
    Case "frmAddRemWshTypes"
        'If g_bStepThruMode Then Stop
        g_cMsgBoxHandler.AskReallyWantToCxlWorkOnFormQues
        ansCloseForm = g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbYesNo)
        If ansCloseForm = vbNo Then GoTo ExitPoint
        Unload m_frmAddRemWshTypes
        Set m_frmAddRemWshTypes = Nothing
    Case "frmTypeWshAssoc"
        If g_bStepThruMode Then Stop
        g_cMsgBoxHandler.AskReallyWantToCxlWorkOnFormQues
        ansCloseForm = g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbYesNo)
        If ansCloseForm = vbNo Then GoTo ExitPoint
        If g_bStepThruMode Then Stop
        Dim WsType As String
        Dim yWshTypes As Byte
        Dim i As Byte
        'Blow out all Worksheet Types that we built...
        yWshTypes = g_cDimHandler.WorksheetTypes.Count
        For i = 1 To yWshTypes
            'Because we are removing, work backwards...
            WsType = g_cDimHandler.WorksheetTypes(1 + yWshTypes - i).name
            g_cDimHandler.WorksheetTypes.Remove _
                    index:=WsType, _
                    RanOkRESULT:=bCalledProcedureRanOk

        Next i
        'Redefine WshTypes collection from source file table...
        g_cDimHandler.CreateWshTypesFromFile RanOkRESULT:=bCalledProcedureRanOk
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        Unload m_frmWshTypesWshAssoc
        Set m_frmWshTypesWshAssoc = Nothing
    Case "frmWshTypePicker"
        Unload m_frmWshTypePicker
        Set m_frmWshTypePicker = Nothing
    Case "frmWshTypeDimens"
        If g_bStepThruMode Then Stop
        g_cMsgBoxHandler.AskReallyWantToCxlWorkOnFormQues
        ansCloseForm = g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbYesNo)
        If ansCloseForm = vbNo Then GoTo ExitPoint
        'Close form without saving data...
        If g_bStepThruMode Then Stop
        Unload Me.WshTypeDimensFormsColl.Item(WshTypeName)
        Me.WshTypeDimensFormsColl.Remove _
                index:=WshTypeName, _
                RanOkRESULT:=bCalledProcedureRanOk
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
End Select

'If the user cancels out of ANY form he's out of the wizard business.  It's _
just too complicated to pick up the wizard where he left off.  Ergo...
g_bWizardMode = False

ThisWorkbook.Activate

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
            ModuleName:=m_sMODULE_NAME, _
```

```
            ProcedureName:=sSOURCE, _
            StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
            If g_bDebugMode Then Stop
            'So we can step through the code which caused the error...
            Resume
        Else
            Resume ExitPoint
        End If
End Sub
'============================================================================
Public Sub PostWshTypesFromForm( _
    ByVal UsingALLType As Boolean, _
    ByVal OnlyOneWshType As Boolean, _
    ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "PostWshTypesFromForm()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Const sFILE_TEXT_REF As String = "File Hard Coding"
Const sFILE_RNG_REF As String = "File Range Ref"
Const yFILE_DIMEN_COLS As Byte = 2
Const yDIMEN_ID_COLS As Byte = 3
Dim wsSrcWbkValList As Worksheet
Dim wsSrcWbkDimLocs As Worksheet
Dim tblDimLoc As ListObject
Dim lcoDimenLoc As ListColumn
Dim rngdoWshTypeTbl As Range
Dim rngdoWshTypeTblRow As Range
Dim rngHdrzDimLocTbl As Range
Dim c As Range
Dim sWshType As String
Dim bAnyWshTypeDimensInXL As Boolean
Dim bDimLocColInWshTypeList As Boolean
Dim bWshTypeInForm As Boolean
Dim bWshTypeInWshTypeTbl As Boolean
Dim bWshTypeInDimLocTbl As Boolean
Dim yWshTypesOnForm As Byte
Dim yWshTypesOnTbl As Byte
Dim yTotalDimLocCols As Byte
Dim yWshTypeDimLocCols As Byte
Dim yRowOffset As Byte
Dim i As Byte
Dim j As Byte

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

If g_bStepThruMode Then Stop

'CODE HERE...
Set wsSrcWbkValList = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
Set wsSrcWbkDimLocs = g_wbSrcData.Worksheets(g_sWSH_DIMEN_LOCS)
Set rngdoWshTypeTbl = wsSrcWbkValList.Range("valtbldodynWshTypeInfo")
Set tblDimLoc = wsSrcWbkDimLocs.ListObjects("tblDimenLocs")
Set rngHdrzDimLocTbl = tblDimLoc.HeaderRowRange
yWshTypesOnForm = m_frmAddRemWshTypes.lstTypes.ListCount
yWshTypesOnTbl = _
    WorksheetFunction.CountA(rngdoWshTypeTbl.Columns(g_yWSH_TYPE_TBL_NM_COL))

'Remove Wsh Types from WshType Info table if the type is not on the form...
For i = 0 To yWshTypesOnTbl - 1
    'Work from the bottom up...
    If g_bStepThruMode Then Stop
    bWshTypeInForm = False '...reset
    Set rngdoWshTypeTblRow = rngdoWshTypeTbl.Rows(yWshTypesOnTbl - i)
    sWshType = Trim(rngdoWshTypeTblRow.Cells(1, g_yWSH_TYPE_TBL_NM_COL))
    For j = 1 To yWshTypesOnForm
```

```
        If StrComp(sWshType, Trim(m_frmAddRemWshTypes.lstTypes.List(j - 1, 0)), _
                vbTextCompare) = 0 Then
            bWshTypeInForm = True
            j = yWshTypesOnForm '...break loop
        End If
    Next j
    If Not bWshTypeInForm Then rngdoWshTypeTblRow.ClearContents
Next i

'Close up table in case we've removed any wsh types...
If yWshTypesOnTbl > 1 Then
    rngdoWshTypeTbl.Sort _
            key1:=rngdoWshTypeTbl.Columns(g_yWSH_TYPE_TBL_DEFINED_COL), _
            order1:=xlAscending, _
            key2:=rngdoWshTypeTbl.Columns(g_yWSH_TYPE_TBL_NM_COL), _
            order2:=xlAscending, _
            Header:=xlNo
    'Redefine range...
    Set rngdoWshTypeTbl = wsSrcWbkValList.Range("valtbldodynWshTypeInfo")
End If

'Now add WshTypes from form if the type is not in the table...
If g_bStepThruMode Then Stop
yWshTypesOnTbl = _
    WorksheetFunction.CountA(rngdoWshTypeTbl.Columns(g_yWSH_TYPE_TBL_NM_COL))
For i = 1 To yWshTypesOnForm
    bWshTypeInWshTypeTbl = False '...reset
    sWshType = Trim(m_frmAddRemWshTypes.lstTypes.List(i - 1, 0))
    If yWshTypesOnTbl > 0 Then
        For j = 1 To yWshTypesOnTbl
            If StrComp(sWshType, _
                    Trim(rngdoWshTypeTbl.Cells(j, g_yWSH_TYPE_TBL_NM_COL)), _
                    vbTextCompare) = 0 Then
                bWshTypeInWshTypeTbl = True
                j = yWshTypesOnTbl '...break loop
            End If
        Next j
    End If
    If Not bWshTypeInWshTypeTbl Then
        'If there's nothing in the table (e.g., because we have deleted the _
        ALL type) and this is the first WshType on the form, then destination _
        range is 1 row and EMPTY.  Ergo, we want to put the WshType IN that _
        row, rather than below it...
        If i = 1 And yWshTypesOnTbl = 0 Then
            yRowOffset = 0
        Else
            yRowOffset = rngdoWshTypeTbl.Rows.Count
        End If
        rngdoWshTypeTbl.Offset(yRowOffset, 0).Cells(1, g_yWSH_TYPE_TBL_NM_COL) = _
            Trim(m_frmAddRemWshTypes.lstTypes.List(i - 1, 0))
        rngdoWshTypeTbl.Offset(yRowOffset, 0).Cells(1, _
                g_yWSH_TYPE_TBL_DEFINED_COL) = False
        'Redefine range...
        Set rngdoWshTypeTbl = wsSrcWbkValList.Range("valtbldodynWshTypeInfo")
    End If
Next i

'If g_bStepThruMode Then Stop
'Sort the little darlings...
rngdoWshTypeTbl.Sort _
        key1:=rngdoWshTypeTbl.Columns(g_yWSH_TYPE_TBL_DEFINED_COL), _
        order1:=xlAscending, _
        key2:=rngdoWshTypeTbl.Columns(g_yWSH_TYPE_TBL_NM_COL), _
        order2:=xlAscending, _
        Header:=xlNo

'We apparently need to unprotect our wsh here to delete table columns...
wsSrcWbkDimLocs.Unprotect g_sWSH_PASSWORD

'Remove any columns from Dimen Locations table that are no longer in the list...
'NOTE:  Because we are deleting columns we want to move from R to L across _
```

```
the table...
yTotalDimLocCols = tblDimLoc.ListColumns.Count
yWshTypeDimLocCols = yTotalDimLocCols - (yFILE_DIMEN_COLS + yDIMEN_ID_COLS)
If yWshTypeDimLocCols = 0 Then GoTo AddColumns
'Loop R to L through each Wsh Type header
'[NOTE:  I originally wrote this as a For/Next using Step -1 but, for some _
reason, the VBA compiler kept choking on the Step -1 part.]...
For i = 1 To yWshTypeDimLocCols
    sWshType = Trim(rngHdrzDimLocTbl.Cells(1, 1 + yTotalDimLocCols - i))
    bDimLocColInWshTypeList = False '...reset
    'Loop through each Wsh Type in wsh list...
    For j = 1 To yWshTypesOnForm
        If StrComp(sWshType, _
                Trim(rngdoWshTypeTbl.Cells(j, g_yWSH_TYPE_TBL_NM_COL)), _
                vbTextCompare) = 0 Then
            bDimLocColInWshTypeList = True
            j = yWshTypesOnForm '...break loop
        End If
    Next j
    If Not bDimLocColInWshTypeList Then tblDimLoc.ListColumns( _
            1 + yTotalDimLocCols - i).Delete
Next i

AddColumns:
yTotalDimLocCols = tblDimLoc.ListColumns.Count '...recount
yWshTypeDimLocCols = yTotalDimLocCols - (yFILE_DIMEN_COLS + yDIMEN_ID_COLS)
If yWshTypeDimLocCols = yWshTypesOnForm Then GoTo OnlyOneType

'Loop through entries on Val List Wsh...
For i = 1 To yWshTypesOnForm '... = # entries in Val List
    sWshType = Trim(rngdoWshTypeTbl.Cells(i, g_yWSH_TYPE_TBL_NM_COL))
    If yWshTypeDimLocCols = 0 Then
        'Nothing to check - EVERY entry in ValList must be added...
        Set lcoDimenLoc = tblDimLoc.ListColumns.Add()
        lcoDimenLoc.name = sWshType
        lcoDimenLoc.DataBodyRange.EntireColumn.ColumnWidth = _
                g_yCOL_WIDTH_DIMEN_LOCS
    Else '...add WshType if not in DimenLoc table
        bWshTypeInDimLocTbl = False '...reset
        'Loop through Wsh Type hdrs in Dimen Loc table...
        For j = yFILE_DIMEN_COLS + 1 + 1 To yTotalDimLocCols
            If StrComp(UCase(sWshType), _
                    UCase(Trim(tblDimLoc.ListColumns(j).name))) = 0 Then
                bWshTypeInDimLocTbl = True
                j = yTotalDimLocCols '...break loop
            End If
        Next j
        If Not bWshTypeInDimLocTbl Then
            Set lcoDimenLoc = tblDimLoc.ListColumns.Add
            lcoDimenLoc.name = sWshType
            lcoDimenLoc.DataBodyRange.EntireColumn.ColumnWidth = _
                g_yCOL_WIDTH_DIMEN_LOCS
            'NOTE:  Do NOT reset yTotalDimLocCols, otherwise we will check _
            columns we are adding!
        End If
    End If
Next i

OnlyOneType:
If OnlyOneWshType Then _
   ForOneWshTypePopWshTypeCollAndWshToWshTypeMap RanOkRESULT:=bCalledProcedureRanOk

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
wsSrcWbkValList.Protect Password:=g_sWSH_PASSWORD, contents:=True, userinterfaceonly:=True
g_wbSrcData.Save
Exit Sub

ErrorHandler:
        RanOkRESULT = False
```

```
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Public Sub OpenAndPopulateAssocWshTypeWithWshForm(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "OpenAndPopulateAssocWshTypeWithWshForm()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Const sSTEP_FILE_TTL As String = "STEP 5 OF 8:  "
Const sWSH_DIMEN_FORM_TTL As String = "Assign Wsh Types with Worksheets"
Dim wsSrcWbkValList As Worksheet
Dim objWshType As WshType
Dim rngdoWshTypeTbl As Range
Dim rngdoDataWshs As Range
Dim sWsh As String
Dim saAllDataWshs() As String
Dim saWshTypes() As String
Dim sFormCaption As String
Dim bIsWshAssignedToAType As Boolean
Dim yAllDataWshs As Byte
Dim yWshTypes As Byte
Dim i As Byte
Dim j As Byte

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Set wsSrcWbkValList = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
Set rngdoWshTypeTbl = wsSrcWbkValList.Range("valtbldodynWshTypeInfo")
Set rngdoDataWshs = wsSrcWbkValList.Range("vallstdodynWshsToImport")

'If g_bStepThruMode Then Stop
'Set up form...
sFormCaption = sWSH_DIMEN_FORM_TTL
If g_bWizardMode Then sFormCaption = sSTEP_FILE_TTL & sFormCaption
Set m_frmWshTypesWshAssoc = _
        g_cObjFactory.CreateTypeWshLinkForm( _
                FormCaption:=sFormCaption, _
                RanOkRESULT:=bCalledProcedureRanOk)
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

If g_bStepThruMode Then Stop
If Not g_bWizardMode Then _
    g_cDimHandler.CreateWshTypesFromFile _
            RanOkRESULT:=bCalledProcedureRanOk
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

If g_bStepThruMode Then Stop
'Because we never know if the user might Cxl out of the form, we don't want _
changes in the form to touch the WshTypesColl in DimensionHandler until the _
user hits OK.  In order to manage things whilst the user associates and _
disassociates worksheets with worksheet types what we do is to create a _
cloned version of the worksheet types collection inside the form.  All of _
the association and disassociation affects that, cloned collection.  If the _
user cancels out of the form our collection in DimenHandler is left untouched. _
However, if the user clicks OK, we replace DimensionHandler's WshTypes _
```

```
FormsHandler - 50

collection with a cloned version of the form's collection...

'Clone WshTypesColl inside form...
m_frmWshTypesWshAssoc.CloneWshTypesColl RanOkRESULT:=bCalledProcedureRanOk
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

'Populate unassigned wsh listbox...
'If g_bStepThruMode Then Stop
For i = 1 To rngdoDataWshs.Rows.Count
    sWsh = Trim(rngdoDataWshs.Cells(i, 1))
    bIsWshAssignedToAType = False '...reset
    'Loop through each worksheet type...
    For j = 1 To g_cDimHandler.WorksheetTypes.Count
        Set objWshType = g_cDimHandler.WorksheetTypes(j)
        If objWshType.ContainsItem(sWsh) Then
            bIsWshAssignedToAType = True
            j = g_cDimHandler.WorksheetTypes.Count '...break loop
        End If
    Next j
    If Not bIsWshAssignedToAType Then _
        m_frmWshTypesWshAssoc.lstUnassgndWshs.AddItem sWsh
Next i

'Populate combo box with Worksheet types...
'If g_bStepThruMode Then Stop
For i = 1 To rngdoWshTypeTbl.Rows.Count
    m_frmWshTypesWshAssoc.cmbWshTypes.AddItem _
            rngdoWshTypeTbl.Cells(i, g_yWSH_TYPE_TBL_NM_COL)
Next i

m_frmWshTypesWshAssoc.Show

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'============================================================================
Public Sub OpenAndPopulateWshTypePickerForm(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "OpenAndPopulateWshTypePickerForm()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
If Not IsFormOpen( _
        FormName:=frmWshTypePicker.name, _
        RanOkRESULT:=bCalledProcedureRanOk) Then
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    Set m_frmWshTypePicker = _
```

```
            g_cObjFactory.CreateWshTypePickerForm(RanOkRESULT:=bCalledProcedureRanOk)
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
Else
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    'Update the list...
    m_frmWshTypePicker.ClearAndRepopulateListBox bCalledProcedureRanOk
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
End If

m_frmWshTypePicker.Show

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'=============================================================================
Public Sub UserSelectWshTypeFromWshTypePickerForm(ByVal WshTypeName As String)
'Error-handling declarations...
Const sSOURCE As String = "UserSelectWshTypeFromWshTypePickerForm()"
'Treat this as an entry point, because we are going to destroy the _
WshTypePicker form, and therefore do not want to pass code back to it...
Const bSUB_IS_ENTRY_PT As Boolean = True
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

On Error GoTo ErrorHandler

'CODE HERE...
If g_bStepThruMode Then Stop
WshTypeName = Trim(WshTypeName)
m_frmWshTypePicker.Hide

OpenWshTypeDimenIDForm _
        OpenedFmAnotherWTDForm:=False, _
        RanOkRESULT:=bCalledProcedureRanOk, _
        WshTypeNm:=WshTypeName

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
```

```
                Resume ExitPoint
        End If
End Sub
'===============================================================================
Public Sub PostWshTypeToWshAssocToSrcDataWbk( _
    ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "PostWshTypeToWshAssocToSrcDataWbk()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim objWsType As WshType
Dim wsValList As Worksheet
Dim rngdoDataWshs As Range
Dim rngdoWshTypeWshNmTbl As Range
Dim rngWshTypeWshNmHdrs As Range
Dim rngWsNameHdr As Range
Dim rngWsTypeHdr As Range
Dim c As Range
Dim sWshTypeNm As String
Dim sEmptyWshTypes() As String
Dim bWshTypeHasNoWshsAssigned As Boolean
Dim yEmptyWshTypes As Byte
Dim yRowNo As Byte
Dim yNoWshs As Byte
Dim i As Byte
Dim j As Byte

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
'Trap for unassigned data worksheets...
If m_frmWshTypesWshAssoc.lstUnassgndWshs.ListCount > 0 Then
    g_cMsgBoxHandler.SetUnassignedWshsMsg
    g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbExclamation
    GoTo ExitPoint
End If

If g_bStepThruMode Then Stop
'Trap for WshTypes that have no worksheets assigned to them...
For i = 1 To m_frmWshTypesWshAssoc.WorksheetTypes.Count
    If m_frmWshTypesWshAssoc.WorksheetTypes(i).Count = 0 Then
        yEmptyWshTypes = yEmptyWshTypes + 1
        ReDim sEmptyWshTypes(0 To yEmptyWshTypes - 1) As String
        sWshTypeNm = m_frmWshTypesWshAssoc.WorksheetTypes(i).name
        sEmptyWshTypes(yEmptyWshTypes - 1) = sWshTypeNm
        bWshTypeHasNoWshsAssigned = True
    End If
Next i
'Notify user...
If bWshTypeHasNoWshsAssigned Then
    g_cMsgBoxHandler.SetEmptyWshTypesMsg sEmptyWshTypes()
    g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbInformation
    GoTo ExitPoint
End If

'***If here we have good data from the user...
'Replace the WshTypes collection in DimensionHandler with the collection _
in the form...
g_cDimHandler.WorksheetTypes = _
        m_frmWshTypesWshAssoc.WorksheetTypes.ICloneable_Clone( _
        RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set wsValList = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
Set rngdoDataWshs = wsValList.Range("vallstdodynWshsToImport")
Set rngdoWshTypeWshNmTbl = wsValList.Range("valtbldodynWshNmTypeMap")
Set rngWshTypeWshNmHdrs = wsValList.Range("valhdrzWshNmTypeMap")
Set rngWsNameHdr = rngWshTypeWshNmHdrs.Cells(1, g_yWSH_ASSOC_TBL_WSH_NM_COL)
```

```
Set rngWsTypeHdr = rngWshTypeWshNmHdrs.Cells(1, g_yWSH_ASSOC_TBL_TYPE_COL)

'Clear out our target range...
rngdoWshTypeWshNmTbl.ClearContents
'Populate table...
yRowNo = 1
For Each objWsType In g_cDimHandler.WorksheetTypes
    yNoWshs = objWsType.Count
    For i = 1 To yNoWshs
        rngWsNameHdr.Offset(yRowNo, 0) = objWsType.Items(i).name
        rngWsTypeHdr.Offset(yRowNo, 0) = objWsType.name
        yRowNo = yRowNo + 1
    Next i
Next objWsType

'Sort the little darlings...
Set rngdoWshTypeWshNmTbl = wsValList.Range("valtbldodynWshNmTypeMap") '...reset
rngdoWshTypeWshNmTbl.Sort _
    key1:=rngdoWshTypeWshNmTbl.Columns(g_yWSH_ASSOC_TBL_WSH_NM_COL), _
    order1:=xlAscending, _
    Header:=xlNo

g_wbSrcData.Save

'Now handle clean-up...
If g_bStepThruMode Then Stop
m_frmWshTypesWshAssoc.Hide
If g_bWizardMode Then
    RibbonHandler.SetNextStageAndResetRibbon projNxtStgWshTypeDimens
    If g_bStepThruMode Then Stop
    g_wbSrcData.Activate
    OpenAndPopulateWshTypePickerForm _
            RanOkRESULT:=bCalledProcedureRanOk
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
Else
    'We don't care here whether the user has deleted or renamed worksheets. _
    All that matters is whether there are any worksheet types with _
    undefined mandatory dimensions...
    If g_cDimHandler.AnyWshTypesWithUndefinedMandDimens( _
            RanOkRESULT:=bCalledProcedureRanOk) Then
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        RibbonHandler.SetNextStageAndResetRibbon projNxtStgWshTypeDimens
    Else
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        RibbonHandler.SetNextStageAndResetRibbon projNxtStgImportData
    End If
End If

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Public Sub AddRemoveWshsToWshTypeAssocAndUpdateListBoxes( _
```

```
    ByVal AssocDisassoc As ProjAssocDisassocWsh, _
    ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "AddRemoveWshsToWshTypeAssocAndUpdateListBoxes()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim lstSourceListBox As MSForms.ListBox
Dim lstDestListBox As MSForms.ListBox
Dim saWshsToAdd() As String
Dim saDestListBoxWshs() As String
Dim sWshType As String
Dim sWshNm As String
Dim iWshTypeIndex As Integer
Dim yDestLstBoxWshs As Byte
Dim ySrcLstBoxWshs As Byte
Dim bUserSelectedAWsh As Boolean

Dim yItemNo As Byte
Dim i As Byte

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'If g_bStepThruMode Then Stop
'CODE HERE...
iWshTypeIndex = m_frmWshTypesWshAssoc.cmbWshTypes.ListIndex
'Trap for user not having selected a worksheet type...
If iWshTypeIndex = -1 Then
    g_cMsgBoxHandler.SetNoWshTypeSelectedMsg
    g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbExclamation
    GoTo ExitPoint
End If

sWshType = m_frmWshTypesWshAssoc.cmbWshTypes.List(iWshTypeIndex, 0)

'Define source and destination list boxes...
Select Case AssocDisassoc
    Case projAssoc
        Set lstSourceListBox = m_frmWshTypesWshAssoc.lstUnassgndWshs
        Set lstDestListBox = m_frmWshTypesWshAssoc.lstAssocWshs
        ySrcLstBoxWshs = lstSourceListBox.ListCount
        yDestLstBoxWshs = lstDestListBox.ListCount
    Case projDisassoc
        Set lstSourceListBox = m_frmWshTypesWshAssoc.lstAssocWshs
        Set lstDestListBox = m_frmWshTypesWshAssoc.lstUnassgndWshs
        ySrcLstBoxWshs = lstSourceListBox.ListCount
        yDestLstBoxWshs = lstDestListBox.ListCount
End Select

'Trap for an empty source list box...
If ySrcLstBoxWshs = 0 Then GoTo ExitPoint

'Loop through and identify selected worksheets (i.e., in source list box)...
For i = 0 To ySrcLstBoxWshs - 1
    'Since I am STILL having problems with STEP -1...
    yItemNo = ySrcLstBoxWshs - i - 1
    If lstSourceListBox.Selected(yItemNo) Then
        bUserSelectedAWsh = True
        sWshNm = lstSourceListBox.List(yItemNo)
        'Add or remove worksheet to/from appropriate WshType object...
        Select Case AssocDisassoc
            Case projAssoc
                m_frmWshTypesWshAssoc.WorksheetTypes(sWshType).Add g_wbSrcData.Worksheets(sWshNm)
            Case projDisassoc
                m_frmWshTypesWshAssoc.WorksheetTypes(sWshType).Remove _
                        index:=sWshNm, _
                        RanOkRESULT:=bCalledProcedureRanOk
                    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        End Select
```

```
        'Add worksheet to appropriate list box...
        lstDestListBox.AddItem sWshNm
        'Remove selected worksheet from appropriate list box...
        lstSourceListBox.RemoveItem yItemNo
    End If
Next i

'Trap for the user not having selected a worksheet in the source list box...
If Not bUserSelectedAWsh Then GoTo ExitPoint

'Sort worksheet names in destination list box (must sort because _
there may well have been worksheets in the list box already)...
yDestLstBoxWshs = lstDestListBox.ListCount '...this variable has changed
ReDim saDestListBoxWshs(0 To yDestLstBoxWshs - 1)
For i = 0 To yDestLstBoxWshs - 1
    saDestListBoxWshs(i) = lstDestListBox.List(i)
Next i
If yDestLstBoxWshs > 1 Then
    Utilities.BubbleSortArray _
            arr:=saDestListBoxWshs, _
            RanOkRESULT:=bCalledProcedureRanOk
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    'Now update dest list box with sorted array...
    lstDestListBox.Clear
    For i = 0 To yDestLstBoxWshs - 1
        lstDestListBox.AddItem saDestListBoxWshs(i)
    Next i
End If

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'=============================================================================
Public Sub UpdateAssociatedWshListBoxForSelectedWshType( _
    ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "UpdateAssociatedWshListBoxForSelectedWshType()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim objWshType As WshType
Dim ws As Worksheet
Dim sWshType As String
Dim iWshTypeIndx As Integer

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
iWshTypeIndx = m_frmWshTypesWshAssoc.cmbWshTypes.ListIndex

'Trap for empty combo box...
```

```
FormsHandler - 56

If iWshTypeIndx = -1 Then GoTo ExitPoint

'Get WorksheetType collection...
sWshType = m_frmWshTypesWshAssoc.cmbWshTypes.List(iWshTypeIndx, 0)
Set objWshType = m_frmWshTypesWshAssoc.WorksheetTypes(sWshType)

'Clear and repopulate list box...
m_frmWshTypesWshAssoc.lstAssocWshs.Clear
For Each ws In objWshType
    m_frmWshTypesWshAssoc.lstAssocWshs.AddItem ws.name
Next ws

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Public Sub PopulateDimenIDFormWithValueFromSrcWbkDimenTbl( _
    ByVal DimScope As ProjDimenScope, _
    ByVal ControlType As ProjCtrlType, _
    ByVal DimenName As String, _
    ByVal DimenValue As String, _
    ByRef RanOkRESULT As Boolean, _
    Optional WshTypeName As String)
'Error-handling declarations...
Const sSOURCE As String = "PopulateDimenIDFormWithValueFromSrcWbkDimenTbl()"
Const bSUB_IS_ENTRY_PT As Boolean = False
'''Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim txtCtrl As MSForms.TextBox
Dim refCtrl As RefEdit.RefEdit
Dim i As Byte

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'If g_bStepThruMode Then Stop
'Just to be sure...
DimenName = Trim(DimenName)
DimenValue = Trim(DimenValue)

'CODE HERE...
Select Case DimScope
    Case projScopeFile
        Select Case ControlType
            Case projCtrlTypTxtBox
                'At the FileDimension level we treat our pseudo-dimensions _
                as text type dimensions...
                If StrComp(DimenName, g_sDIMEN_SCALE, vbTextCompare) = 0 Then
                    For i = 0 To g_ySCALE_OPTIONS - 1
                        If StrComp(m_frmFileMandDimens.cmbScale.List(i, 0), _
                                DimenValue, vbTextCompare) = 0 Then
                            m_frmFileMandDimens.cmbScale.ListIndex = i
```

```
                        i = g_ySCALE_OPTIONS - 1 '...break loop
                    End If
                Next i
            Else
                For i = 1 To m_frmFileMandDimens.TextBoxesByCtrlNm.Count
                    Set txtCtrl = m_frmFileMandDimens.TextBoxesByCtrlNm(i)
                    If StrComp(txtCtrl.Tag, DimenName) = 0 Then
                        'If g_bStepThruMode Then Stop
                        txtCtrl.value = DimenValue
                        i = m_frmFileMandDimens.TextBoxesByCtrlNm.Count  '...break loop
                    End If
                Next i
            End If
        Case projCtrlTypeRefEdit
            For i = 1 To m_frmFileMandDimens.RefEditsByCtrlNm.Count
                Set refCtrl = m_frmFileMandDimens.RefEditsByCtrlNm(i)
                If StrComp(refCtrl.Tag, DimenName) = 0 Then
                    'If g_bStepThruMode Then Stop
                    refCtrl.value = DimenValue
                    i = m_frmFileMandDimens.RefEditsByCtrlNm.Count - 1 '...break loop
                End If
            Next i
    End Select
Case Else
    Dim wtdForm As frmWshTypeDimens
    Set wtdForm = Me.WshTypeDimensFormsColl.Item(WshTypeName)
    Select Case DimScope
        Case projScopeWsh
            'We treat Amt Scale as a Wsh-wide dimension...
            If StrComp(DimenName, g_sDIMEN_SCALE, vbTextCompare) = 0 Then
                For i = 0 To g_ySCALE_OPTIONS - 1
                    If StrComp(Trim(wtdForm.cmbScale.List(i, 0)), _
                            DimenName, vbTextCompare) Then
                        wtdForm.cmbScale.ListIndex = i
                        i = g_ySCALE_OPTIONS - 1 '...break loop
                    End If
                Next i
            Else
                'All regular dimensions...
                wtdForm.WshDimenRefEditsByDimenNm.Item(DimenName) = DimenValue
            End If
        Case projScopeRow
            wtdForm.RowDimenRefEditsByDimenNm.Item(DimenName) = DimenValue
        Case projScopeCol
            wtdForm.ColDimenRefEditsByDimenNm.Item(DimenName) = DimenValue
    End Select
End Select

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
            ModuleName:=m_sMODULE_NAME, _
            ProcedureName:=sSOURCE, _
            StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
            If g_bDebugMode Then Stop
            'So we can step through the code which caused the error...
            Resume
    Else
            Resume ExitPoint
    End If
End Sub
'================================================================================
Public Sub PopulateDimensOnOpenCollectionInWbkDimenForm( _
```

```
    ByRef RanOkRESULT As Boolean, _
    Optional ScaleSet As Boolean, _
    Optional CurrSet As Boolean, _
    Optional Dimen As Dimension)
'Error-handling declarations...
Const sSOURCE As String = "PopulateDimensOnOpenCollectionInWbkDimenForm()"
Const bSUB_IS_ENTRY_PT As Boolean = False
'Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
If ScaleSet Then
    m_frmFileMandDimens.ScaleSetOnOpen = True
ElseIf CurrSet Then
    m_frmFileMandDimens.CurrTypeSetOnOpen = True
Else
    m_frmFileMandDimens.DimensPopulatedOnOpen.Add Dimen
End If

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
            ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'================================================================================
Public Sub PopulateWTDFormWithFileDimenAndDisableCtrls( _
    ByVal WshTypeName As String, _
    ByVal DimenNm As String, _
    ByVal DimenValue As String, _
    ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = _
    "PopulateWTDFormWithFileDimenAndDisableCtrls()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim frmWTD As frmWshTypeDimens
Dim lblFileDimen As MSForms.label
Dim refWshDimen As RefEdit.RefEdit
Dim refColDimen As RefEdit.RefEdit
Dim refRowDimen As RefEdit.RefEdit
Dim cmbSelectedScale As MSForms.ComboBox

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Set frmWTD = Me.WshTypeDimensFormsColl(WshTypeName)

Set lblFileDimen = frmWTD.FileDimenLblsByDimenNm.Item(DimenNm)
```

```
FormsHandler - 59

lblFileDimen.Caption = DimenValue

'Trap for our pseudo-dimension(s)...
If StrComp(DimenNm, g_sDIMEN_SCALE) = 0 Then
    Set cmbSelectedScale = frmWTD.cmbScale
    With cmbSelectedScale
        .enabled = False
        .BackColor = g_lFIELD_COLOR_DISABLED
        .ListIndex = -1
    End With
Else
    Set refWshDimen = _
            frmWTD.WshDimenRefEditsByDimenNm.Item(DimenNm)
    With refWshDimen
        .enabled = False
        .BackColor = g_lFIELD_COLOR_DISABLED
        .value = vbNullString
    End With

    Set refRowDimen = _
            frmWTD.RowDimenRefEditsByDimenNm.Item(DimenNm)
    With refRowDimen
        .enabled = False
        .BackColor = g_lFIELD_COLOR_DISABLED
        .value = vbNullString
    End With

    Set refColDimen = _
            frmWTD.ColDimenRefEditsByDimenNm.Item(DimenNm)
    With refColDimen
        .enabled = False
        .BackColor = g_lFIELD_COLOR_DISABLED
        .value = vbNullString
    End With
End If

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'============================================================================
Public Sub SelectWshsToImportFromSrcWbkValList(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "SelectWshsToImportFromSrcWbkValList()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wsValLists As Worksheet
Dim rngdoSrcWbkWshsToImport As Range
Dim c As Range
Dim yDataWshs As Byte
Dim bWshToImport As Boolean
Dim i As Byte
Dim j As Byte
```

```
FormsHandler - 60


On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Set wsValLists = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
Set rngdoSrcWbkWshsToImport = wsValLists.Range("vallstdodynWshsToImport")

'Trap for an empty list...
If StrComp(Trim(rngdoSrcWbkWshsToImport.Cells(1, 1)), vbNullString) = 0 Then _
    GoTo ExitPoint

For i = 0 To m_frmImportWshs.lstWshsToImport.ListCount - 1
    For j = 1 To rngdoSrcWbkWshsToImport.Rows.Count
        If StrComp(Trim(m_frmImportWshs.lstWshsToImport.List(i, 0)), _
                Trim(rngdoSrcWbkWshsToImport.Cells(j, 1))) = 0 Then
            m_frmImportWshs.lstWshsToImport.Selected(i) = True
            j = rngdoSrcWbkWshsToImport.Rows.Count '...break loop
        End If
    Next j
Next i

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Public Sub UserOKClosesWshDimenFormMaestro( _
        ByVal WshTypeName As String, _
        ByVal AllMandDimensDefined As Boolean, _
        ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "UserOKClosesWshDimenFormMaestro()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wsValLists As Worksheet
Dim rngWshTypeInfoTbl As Range

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
WshTypeName = Trim(WshTypeName)
If g_bStepThruMode Then Stop
'Set MandDimens Defined flag in XL...
g_cDimHandler.SetWshTypeDimensDefinedFlagInWshTypeTable _
        WshTypeName:=WshTypeName, _
        AllMandDimsDefined:=AllMandDimensDefined, _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
g_wbSrcData.Save
```

```
Unload Me.WshTypeDimensFormsColl.Item(WshTypeName)
Me.WshTypeDimensFormsColl.Remove WshTypeName, bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

'Update the combo box on any open forms...
If Me.WshTypeDimensFormsColl.Count > 0 Then
    Dim wtdForm As frmWshTypeDimens
    Dim cmbWshTypes As MSForms.ComboBox
    For Each wtdForm In Me.WshTypeDimensFormsColl
        wtdForm.cmbWshType.Clear
        wtdForm.PopulateWshTypeComboBox RanOkRESULT:=bCalledProcedureRanOk
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    Next wtdForm
End If

If g_bStepThruMode Then Stop
'If there are more than 1 WshType then we used the WshTypePicker form.  We _
would then have to reset it...
Set wsValLists = g_wbSrcData.Worksheets(g_sWSH_VAL_LISTS)
Set rngWshTypeInfoTbl = wsValLists.Range("vallstdodynWshTypes")
If rngWshTypeInfoTbl.Rows.Count > 1 Then
    m_frmWshTypePicker.ClearAndRepopulateListBox _
            RanOkRESULT:=bCalledProcedureRanOk
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
End If

'If all WshTypes have all mandatory dimens defined reset ribbon...
If g_cDimHandler.AnyWshTypesWithUndefinedMandDimens( _
        RanOkRESULT:=bCalledProcedureRanOk) Then
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    RibbonHandler.SetNextStage projNxtStgWshTypeDimens
Else
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    RibbonHandler.SetNextStage projNxtStgImportData
End If
RibbonHandler.ResetRibbon
If g_bStepThruMode Then Stop


ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'================================================================================
Public Sub UnloadForm( _
        ByVal FormName As String, _
        ByRef RanOkRESULT As Boolean, _
        Optional WshTypeNm As String)
'Error-handling declarations...
Const sSOURCE As String = "UnloadForm()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
```

```
On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Select Case FormName
    Case frmWbkMandDimens.name
        Unload m_frmFileMandDimens
        Set m_frmFileMandDimens = Nothing
    Case frmWshsToImport.name
        Unload m_frmImportWshs
        Set m_frmImportWshs = Nothing
    Case frmAddRemWshTypes.name
        Unload m_frmAddRemWshTypes
        Set m_frmAddRemWshTypes = Nothing
    Case frmTypeWshAssoc.name
        Unload m_frmWshTypesWshAssoc
    Case frmWshTypeDimens.name
        Dim frmWTD As frmWshTypeDimens
        Set frmWTD = Me.WshTypeDimensFormsColl.Item(WshTypeNm)
        Me.WshTypeDimensFormsColl.Remove _
                index:=WshTypeNm, _
                RanOkRESULT:=bCalledProcedureRanOk
            If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
        Unload frmWTD
    Case frmWshTypePicker.name
        Unload m_frmWshTypePicker
        Set m_frmWshTypePicker = Nothing
    Case Else
        'No such wsh, so...
        Err.Raise g_lERR_MISMATCHED_WSHS
End Select

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
            ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub


'================================================================================
'================================================================================
'================================================================================
'================================================================================
'================================================================================
'================================================================================
```

ICloneable - 1

```
Public Function clone(ByRef RanOkRESULT As Boolean) As Object

End Function
```

```
MsgBoxHandler - 1

'.....................................................................................
'.....................................................................................
'.......        Developer:              William H. White (consultant)
'.......        With:                   TEKsystems Inc.
'.......                                www.teksystems.com
'.......        For:                    AIG
'.......                                Financial Information Systems
'.......                                1 NY Plaza, 15th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         8/8/2013
'.......        Proj finished:          11/21/2013
'.......        Proj revised:           1/15/2014
'.....................................................................................
'.....................................................................................

'.....................................................................................
'.........................  Class-level COMMENTS  .....................................
'1)  This class is designed to hold all of the clumsy string concatenations _
involved in message box prompts.
'.....................................................................................
'.....................................................................................


'*************************************************************************************
'*************************************************************************************
'*************************  CLASS-LEVEL DECLARATIONS  ************************************
'*************************************************************************************
'*************************************************************************************
Option Explicit
Option Compare Binary '...use cAsE senSITive string comparisons




'*************************************************************************************
'*************************************************************************************
'*******************************  PRIVATE MEMBERS  ***********************************
'*************************************************************************************
'*************************************************************************************

'**************************************
'MODULE/CLASS-WIDE CONSTANTS
'**************************************
'Strings...
Private Const m_sMODULE_NAME As String = "MsgBoxHandler"
Private Const m_sDEV_MSGBOX_TTL As String = "*** DEVELOPMENT-ONLY MESSAGE!!! ***"
Private Const m_sOTHER_POSSIBLE_PROBS_1 As String = _
    "There may be other problem entries on this form.  "
Private Const m_sOTHER_POSSIBLE_PROBS_2 As String = _
    "Unfortunately, we can only notify you of them one by one."


'**************************************
'MODULE/CLASS-WIDE VARIABLES
'**************************************
'Strings...
Private m_sTitle As String
Private m_sPrompt As String
Private m_s2Crs  As String
Private m_s3Crs  As String
Private m_s2Crs1Tab As String
Private m_sCrTab As String
Private m_sCr2Tabs As String
Private m_s2Tabs As String
```

```
MsgBoxHandler - 2


'=============================================================
'==================  (Private) PROCEDURES  ==================
'=============================================================
'=============================================================================
Private Sub Class_Initialize()
m_s2Crs = vbCr & vbCr
m_s3Crs = vbCr & vbCr & vbCr
m_s2Crs1Tab = vbCr & vbCr & vbTab
m_sCrTab = vbCr & vbTab
m_sCr2Tabs = vbCr & vbTab & vbTab
m_s2Tabs = vbTab & vbTab
End Sub



'*****************************************************************************
'*****************************************************************************
'*****************************  PUBLIC MEMBERS  ******************************
'*****************************************************************************
'*****************************************************************************


'=============================================================
'==================  (Public) FUNCTIONS  ==================
'=============================================================
'=============================================================================
Public Function ShowMsgBoxReturnAnswer(MsgBoxStyle As VbMsgBoxStyle) _
    As VbMsgBoxResult
ShowMsgBoxReturnAnswer = MsgBox(m_sPrompt, MsgBoxStyle, m_sTitle)
'Reset matters...
m_sPrompt = vbNullString
m_sTitle = vbNullString
End Function



'=============================================================
'==================  (Public) PROCEDURES  ==================
'=============================================================

'**************************************
'USER MESSAGES
'**************************************
'=============================================================================
Public Sub SetSourceFileClosedMsg()
m_sTitle = "Source File Closed"
m_sPrompt = "It appears that the source file you had identified:  " & _
m_s2Crs1Tab & g_sSrcWbkFullNm & m_s2Crs & _
"Is no longer open." & m_s2Crs & _
"You will need to restart the entire process of opening a source file " & _
"and identifying file dimensions, sheet dimensions, etc."
End Sub
'=============================================================================
Public Sub SetDecideOnHiddenWshsMsg()
m_sTitle = "Closing Source Workbook"
m_sPrompt = "We cannot process your source workbook until you have " & _
"decided how to handle the hidden data worksheets."
End Sub
'=============================================================================
Public Sub SetDimensionAlreadyUsedMsg(ByVal Dimension As String)
m_sTitle = "Dimension Already Used"
m_sPrompt = "The value for the following dimension has already been " & _
"at another (higher) level:" & _
m_s2Crs1Tab & Dimension
End Sub
'=============================================================================
Public Sub SetDimenRngLargerThanOneCellMsg(ByVal Dimension As String, _
    ByVal RangeAddress As String)
m_sTitle = "Range Specified Larger Than One Cell"
```

```
m_sPrompt = "In setting the for the following dimension:" & _
m_s2Crs1Tab & Dimension & m_s2Crs & _
"you identified the following range:" & _
m_s2Crs1Tab & RangeAddress & m_s3Crs & _
"Please select single cells ONLY when specifying file or worksheet " & _
"dimensions."
End Sub
'================================================================================
Public Sub SetActivateSourceWshCellMsg(ByVal SourceWbkName As String)
m_sTitle = "Select Source Worksheet"
m_sPrompt = "Please select your source tab/sheet in your source file:" & _
m_s2Crs1Tab & SourceWbkName & m_s3Crs & _
"When done return to this file and re-click the " & Chr(34) & _
"Set Worksheet Dimensions button." & Chr(34)
End Sub
'================================================================================
Public Sub SetWillAddSLWshsToSrcWbkMsg( _
    ByVal WshNm1 As String, ByVal WshNm2 As String)
m_sTitle = "Adding Two Worksheets to Source Workbook"
m_sPrompt = _
    "We are about to add two worksheets to your source file:" & _
m_s2Crs1Tab & g_sSrcWbkNm & m_s3Crs & _
"The names of these worksheets are:" & m_s2Crs1Tab & _
WshNm1 & m_sCrTab & WshNm2 & m_s2Crs & _
"These tabs/worksheets, plus approximately three dozen named ranges, will " & _
"be saved with your workbook for future use " & _
"by this macro.  No macros/VBA code, however, will be created in your " & _
"source file." & m_s2Crs & "The above worksheets will be visible to you, " & _
"though they will be password-protected and locked."
End Sub
'================================================================================
Public Sub SetYouLostOneOfOurWshsMsg( _
    ByVal WshMissing As String, ByVal WshFound As String)
m_sTitle = g_sSrcWbkNm & " Missing One of Our Control Worksheets"
m_sPrompt = "We added two " & Chr(34) & "Control" & Chr(34) & Chr(32) & _
"tabs/worksheets to your data workbook the last time you ran this " & _
"macro.  Since then, the following worksheet:" & m_s2Crs1Tab & _
WshMissing & m_s2Crs & _
"has either been deleted or renamed." & m_s2Crs & _
"We are going to add that worksheet back to your workbook.  However, " & _
"we also must clear out all entries in the remaining " & _
"worksheet:" & m_s2Crs1Tab & _
WshFound & m_s2Crs & _
"You will now have to re-specify all of your file dimensions, " & _
"worksheet dimensions, etc. as if you were using the macro on your" & _
"data file for the first time."
End Sub
'================================================================================
Public Sub SetInvalidRangeRefMsg(ByVal ProblemField As String)
m_sTitle = "Invalid Range Reference"
m_sPrompt = "You have an invalid range reference for:" & m_s2Crs1Tab & _
ProblemField & m_s2Crs & _
"NOTE:  " & m_sOTHER_POSSIBLE_PROBS_1 & m_sOTHER_POSSIBLE_PROBS_2
End Sub
'================================================================================
Public Sub SetMultiCellRangeRefMsg(ByVal ProblemField As String)
m_sTitle = "Multi-Cell Range Reference"
m_sPrompt = _
    "You have highlighted more than one cell as your range reference for:" & _
    m_s2Crs1Tab & ProblemField & m_s2Crs & _
"NOTE:  " & m_sOTHER_POSSIBLE_PROBS_1 & m_sOTHER_POSSIBLE_PROBS_2
End Sub
'================================================================================
Public Sub SetMultiRowColRangeRefMsg(ByVal DimScope As ProjDimenScope, _
    ByVal ProblemField As String)
Dim sRngType As String
Dim sDimType As String
Select Case DimScope
    Case projScopeRow
        m_sTitle = "Invalid Column Reference"
        sRngType = " column "
```

```
        sDimType = " row labels "
    Case projScopeCol
        m_sTitle = "Invalid Row Reference"
        sRngType = " row "
        sDimType = " column headers "
End Select
m_sPrompt = "You have highlighted more than one" & sRngType & _
    "in identifying the" & sDimType & "for the following dimension:" & _
    m_s2Crs1Tab & ProblemField & m_s2Crs & _
"NOTE:  " & m_sOTHER_POSSIBLE_PROBS_1 & m_sOTHER_POSSIBLE_PROBS_2
End Sub
'================================================================================
Public Sub SetTooManyFileDimensMsg()
m_sTitle = "Too Many Dimenesions Specified"
m_sPrompt = "For each worksheet/worksheet type you will need to specify " & _
"at least one column of row dimension labels and at least one row of " & _
"column dimension headers.  You will not be able to do that because " & _
"you have specified too many dimensions at the file level."
End Sub
'================================================================================
Public Sub SetNoDataWshsSelectedMsg()
m_sTitle = "No Data Worksheets Selected"
m_sPrompt = "You have not identified any worksheets to import." & m_s2Crs & _
"NOTE:  " & m_sOTHER_POSSIBLE_PROBS_1 & m_sOTHER_POSSIBLE_PROBS_2
End Sub
'================================================================================
Public Sub SetDupeDimensionDefinitionsMsg(ByVal DimenNm As String)
m_sTitle = "Duplicate Dimension Definitions"
m_sPrompt = "You have more than one defintion for the following " & _
"dimension:" & m_s2Crs1Tab & DimenNm & m_s2Crs & _
"NOTE:  " & m_sOTHER_POSSIBLE_PROBS_1 & m_sOTHER_POSSIBLE_PROBS_2
End Sub
'================================================================================
Public Sub SetNoWshTypeSelectedMsg()
m_sTitle = "No Worksheet Type Selected"
m_sPrompt = "You must select a worksheet type with which to associate " & _
"worksheets."
End Sub
'================================================================================
Public Sub SetNoWshTypesIdentifiedMsg()
m_sTitle = "No Data Worksheets Types Identified"
m_sPrompt = "You must identify at least one type of worksheet." & m_s2Crs & _
"NOTE:  You are certainly welcome to use " & Chr(34) & "ALL" & Chr(34) & _
" as a worksheet type."
End Sub
'================================================================================
Public Sub SetNewDataWshsAddedMsg(ByRef NewDataWshs() As String)
Const sTOO_MANY_WSHS As String = "Too many to list here..."
Dim sNewDataWshTbl As String
Dim sPrmpt1 As String
Dim sPrmpt2 As String
Dim iNewDataWshs As Integer
Dim i As Byte

m_sTitle = "New Data Worksheets"

iNewDataWshs = UBound(NewDataWshs) + 1
sPrmpt1 = "You have identified the following worksheet(s) for the first " & _
        "time as containing data for the CART conversion process:"
If iNewDataWshs > 10 Then
    sNewDataWshTbl = m_sCrTab & sTOO_MANY_WSHS
Else
    For i = 1 To iNewDataWshs
        sNewDataWshTbl = sNewDataWshTbl & m_sCrTab & NewDataWshs(i - 1)
    Next i
End If
sPrmpt2 = "This is just a reminder that you will not be able to import " & _
        "data from your source workbook until you assign each new " & _
        "worksheet to a Worksheet Type."
m_sPrompt = sPrmpt1 & vbCr & sNewDataWshTbl & m_s2Crs & sPrmpt2
End Sub
```

```
'================================================================================
Public Sub SetImportCARTDataInfoMsg()
m_sTitle = "Just as an FYI..."
m_sPrompt = "In order to speed the process we are freezing your Excel " & _
"screen.  We do, however, provide progress updates in the Excel status " & _
"bar (i.e., the bottom edge of the Excel window)." & m_s2Crs & _
"The import process may take a little while.  We will " & _
"notify you with another message box when the data have been imported."
End Sub
'================================================================================
Public Sub SetDataImportCompleteMsg()
m_sTitle = "Data Import Complete"
m_sPrompt = "All data has been successfully imported into this staging " & _
"layer file, which has now been saved using a modified version of " & _
"the name of your source file." & m_s2Crs & _
"NOTE:  Any worksheet with problem data has been copied to this workbook and " & _
"assigned a " & Chr(34) & "_BAD" & Chr(34) & " suffix.  Missing row " & _
"labels and column headings, and any non-numeric cell entries, have " & _
"been identified with a pure black fill."
End Sub
'================================================================================
Public Sub AskWantToRemoveWshsTypesBcAddingAllTypeQuest( _
    ByVal IncludeWillLoseDimenDefsReminder As Boolean)
Dim sPr1 As String
Dim sPr2 As String
Dim sPr3 As String

m_sTitle = "Remove All Other Worksheet Types?"
sPr1 = "If you add the " & Chr(34) & "ALL" & Chr(34) & _
" type we will remove the other worksheet types you have defined.  "
sPr2 = "Bear in mind that this means that you will lose any and all " & _
"dimension definitions that you had set for those worksheet types!!!"
sPr3 = "Is this what you want to do?"

If IncludeWillLoseDimenDefsReminder Then
    m_sPrompt = sPr1 & m_s2Crs & sPr3
Else
    m_sPrompt = sPr1 & m_s2Crs & UCase(sPr2) & m_s2Crs & sPr3
End If
End Sub
'================================================================================
Public Sub AskWantToRemoveTheAllWshTypeQuest( _
        ByVal IncludeWillLoseDimenDefsReminder As Boolean)
Dim sPr1 As String
Dim sPr2 As String
Dim sPr3 As String

m_sTitle = "Remove the " & Chr(34) & "ALL" & Chr(34) & " Worksheet Type?"
sPr1 = "Adding a worksheet type will automatically remove the  " & _
Chr(34) & "ALL" & Chr(34) & " type."
sPr2 = "Bear in mind that this means you will lose all dimension " & _
        "definitions you had specified for the " & Chr(34) & "ALL" & _
        Chr(34) & " type!!!"
sPr3 = "Is this what you want to do?"
If IncludeWillLoseDimenDefsReminder Then
    m_sPrompt = sPr1 & m_s2Crs & UCase(sPr2) & m_s2Crs & sPr3
Else
    m_sPrompt = sPr1 & m_s2Crs & sPr3
End If
End Sub
'================================================================================
Public Sub AskAddWshTypeEvenThoMustSpecifyDimensQuest()
m_sTitle = "Add a New Worksheet Type?"
m_sPrompt = "Just a reminder...  When you add a new Worksheet Type you " & _
"will not be able to import any data until you have:" & m_s2Crs1Tab & _
"1) Assigned at least one worksheet to the new Worksheet Type" & m_sCrTab & _
"2) Specified all manadatory dimensions for the new Worksheet Type" & _
m_s2Crs & "Do you still want to proceed?"
End Sub
'================================================================================
Public Sub AskWantToIncludeTypeInTxtBoxQuest(ByVal WshTypeNm As String)
```

```
MsgBoxHandler - 6

m_sTitle = "Include " & WshTypeNm & " As a Worksheet Type?"
m_sPrompt = "You have a worksheet type still sitting in the " & _
Chr(34) & "New worksheet type" & Chr(34) & " box." & m_s2Crs & _
"Do you want to create that as a worksheet type?"
End Sub
'================================================================================
Public Sub AskConfirmWantToRemoveWshTypeQuest()
Dim sPr1 As String
Dim sPr2 As String
m_sTitle = "Remove Worksheet Type(s)?"
sPr1 = "When you remove a Worksheet Type you lose all dimension " & _
"definitions you may have established for it!!!"
sPr2 = "Are you sure you want to remove this/these Worksheet Type(s)?"
m_sPrompt = "Just a reminder...  " & m_s2Crs & UCase(sPr1) & m_s2Crs & sPr2
End Sub
'================================================================================
Public Sub AskCloseWshTypePickerFormEvenThoNoWshTypeSelectedQuest()
m_sTitle = "Close This Form?"
m_sPrompt = "You have not selected a worksheet type to work on.  Do you " & _
"still want to close this form?"
End Sub
'================================================================================
Public Sub AskUnhideDataWshsQuest(ByRef HiddenDataWshs() As String)
Dim sPr1 As String
Dim sPr2 As String
Dim sPr3 As String
Dim sPrN As String
Dim yHiddenWshs As Byte
Dim i As Byte

yHiddenWshs = UBound(HiddenDataWshs) + 1
m_sTitle = "Unhide Data Worksheets?"
sPr1 = "Since you last imported data from your source file you have " & _
        "hidden the following data worksheet(s):"
For i = 1 To yHiddenWshs
    sPr2 = m_sCrTab & HiddenDataWshs(i - 1)
Next i
sPr3 = "Do you want to continue to import data from these worksheets?"
sPrN = _
        "If you answer YES..." & m_sCrTab & _
        "We will unhide the above worksheets and save this change " & _
        "to your data file." & m_s2Crs & _
        "If you answer NO..." & m_sCrTab & "We will " & _
        "remove them from the list of worksheets-to-import " & _
        "and save this change to your data file."
m_sPrompt = sPr1 & vbCr & sPr2 & m_s2Crs & sPr3 & m_s3Crs & sPrN
End Sub
'================================================================================
Public Sub AskAddNewFileDimensionsQuest(ByRef NewFileDimens() As String)
Dim sNewDimensTbl As String
Dim sPrompt1 As String
Dim sPrompt2 As String
Dim yNewDimens As Byte
Dim i As Byte

m_sTitle = "Add New File Dimensions?"
sPrompt1 = "You have specified values for dimensions that you had not " & _
        "previously specified at the file level: " & vbCr

yNewDimens = UBound(NewFileDimens) + 1
For i = 1 To yNewDimens
    sNewDimensTbl = _
            sNewDimensTbl & m_sCrTab & NewFileDimens(i - 1)
Next i

sPrompt2 = "Continuing means we will remove all worksheet-type " & _
        "definitions for these dimensions." & m_s2Crs & _
        "Are you sure you want to proceed?"

m_sPrompt = sPrompt1 & sNewDimensTbl & m_s2Crs & sPrompt2
End Sub
```

```
MsgBoxHandler - 7

'================================================================================
Public Sub AskRemoveFileDimensionsQuest(ByRef RemovedFileDimens() As String)
Dim sRemovedDimensTbl As String
Dim sPrompt1 As String
Dim sPrompt2 As String
Dim yRemvdDimens As Byte
Dim i As Byte

m_sTitle = "Remove File Dimensions?"
sPrompt1 = "You have removed values for the following dimension(s):" & vbCr

yRemvdDimens = UBound(RemovedFileDimens) + 1
For i = 1 To yRemvdDimens
    sRemovedDimensTbl = _
            sRemovedDimensTbl & m_sCrTab & RemovedFileDimens(i - 1)
Next i

sPrompt2 = "Removing values for these dimensions " & _
    "will require you to supply values at the Worksheet Type level " & _
    "before importing your source data." & m_s2Crs & _
    "Are you sure you want to proceed?"

m_sPrompt = sPrompt1 & sRemovedDimensTbl & m_s2Crs & sPrompt2
End Sub
'================================================================================
Public Sub AskAddNewDataWshsToExistingWshTypeQuest( _
        ByVal SingleWshType As String, _
        ByRef NewDataWshs() As String)
Const sTOO_MANY_WSHS As String = "Too many to list here..."
Dim sNewDataWshTbl As String
Dim sPrmpt1 As String
Dim sPrmpt2 As String
Dim iNewDataWshs As Integer
Dim i As Byte

m_sTitle = _
        "Add New Data Worksheets to " & UCase(SingleWshType) & _
        " Worksheet Type?"

iNewDataWshs = UBound(NewDataWshs) + 1
sPrmpt1 = "You have identified the following worksheet(s) for the first " & _
        "time as containing data for the CART conversion process:"
If iNewDataWshs > 10 Then
    sNewDataWshTbl = m_sCrTab & sTOO_MANY_WSHS
Else
    For i = 1 To iNewDataWshs
        sNewDataWshTbl = sNewDataWshTbl & m_sCrTab & NewDataWshs(i - 1)
    Next i
End If
sPrmpt2 = "Do you want to include these worksheets as part of the " & _
        UCase(SingleWshType) & " worksheet type?"

m_sPrompt = sPrmpt1 & vbCr & sNewDataWshTbl & m_s2Crs & sPrmpt2
End Sub
'================================================================================
Public Sub AskDeSelectDataWshsEvenThoughEmptyWshTypesWillBeDeletedQuest( _
    ByVal DeselectedWshs As Collection, _
    ByRef ToBeEmptyWshTypes As Collection)
Const sTOO_MANY As String = "Too many to list here..."
Dim sPrompt1 As String
Dim sPrompt2 As String
Dim sPrompt3 As String
Dim sDeselectedWshsTbl As String
Dim sToBeEmptyWshTypesTbl As String
Dim iDeselectedWshs As Integer
Dim yToBeEmptyWshTypes As Byte
Dim i As Integer

m_sTitle = _
    "De-Select Data Worksheets Even Though WorksheetTypes Will Be Removed?"
```

```
MsgBoxHandler - 8

iDeselectedWshs = DeselectedWshs.Count
yToBeEmptyWshTypes = ToBeEmptyWshTypes.Count

If iDeselectedWshs > 5 Then
    sDeselectedWshsTbl = m_sCrTab & sTOO_MANY
Else
    For i = 1 To iDeselectedWshs
        sDeselectedWshsTbl = sDeselectedWshsTbl & m_sCrTab & DeselectedWshs.Item(i)
    Next i
End If

If yToBeEmptyWshTypes > 5 Then
    sToBeEmptyWshTypesTbl = m_sCrTab & sTOO_MANY
Else
    For i = 1 To yToBeEmptyWshTypes
        sToBeEmptyWshTypesTbl = _
            sToBeEmptyWshTypesTbl & m_sCrTab & ToBeEmptyWshTypes.Item(i)
    Next i
End If

sPrompt1 = "You have de-selected the following data worksheet(s):"

sPrompt2 = "If you proceed the following Worksheet Type(s) will be left " & _
        "without any associated worksheets:"

sPrompt3 = "In turn, these Worksheet Types - and all of their associated " & _
"dimension defintions - will be deleted." & m_s3Crs & _
"Do you want to proceed with de-selecting the above worksheets and with " & _
"deleting their associated Worksheet Type(s)?"

m_sPrompt = sPrompt1 & vbCr & sDeselectedWshsTbl & m_s2Crs & _
        sPrompt2 & vbCr & sToBeEmptyWshTypesTbl & m_s2Crs & sPrompt3

End Sub
'================================================================================
Public Sub AskCloseEvenThoUndefinedMandDimensQuest( _
        ByVal UndefDimensTbl As String)
m_sTitle = "Close This Form?"
m_sPrompt = "You have left following mandatory dimension(s) undefined:" & _
m_s2Crs & UndefDimensTbl & vbCr & _
"Do you still want to close this form?"
End Sub
'================================================================================
Public Sub AskImportDataQuest()
m_sTitle = "Import Data?"
m_sPrompt = "Do you want to proceed with importing all specified data " & _
"from the source workbook?"
End Sub
'================================================================================
Public Sub AskExportToCSVQuest()
m_sTitle = "Create *.csv File?"
m_sPrompt = "Do you want to create a *.csv file from your CART data?"
End Sub
'================================================================================
Public Sub SetExportSuccesfulMsg()
m_sTitle = "Data Exported"
m_sPrompt = "You have successfully exported the CART data to a *.csv file."
End Sub
'================================================================================
Public Sub ShowMsgBoxNoAnswer(MsgBoxStyle As VbMsgBoxStyle)
MsgBox m_sPrompt, MsgBoxStyle, m_sTitle
'Reset matters...
m_sPrompt = vbNullString
m_sTitle = vbNullString
End Sub
'================================================================================
Public Sub SetWillLetYouKnowWhenDonePrintingMsg()
m_sTitle = "FYI..."
m_sPrompt = _
    "We will let you know when the printing has been completed."
End Sub
```

```
'================================================================================
Public Sub SetPlsUseOKCxlBtnsMsg()
m_sTitle = "X-Close Control Disabled"
m_sPrompt = "Please use either the OK or Cancel button to close this form."
End Sub
'================================================================================
Public Sub AskReallyWantToCxlWorkOnFormQues()
m_sTitle = "Close This Form?"
m_sPrompt = "You are about to lose any and all CHANGES that you have made " & _
"on this form.  (Original data will NOT be affected.)" & m_s2Crs & _
"Are you sure this is what you want to do?"
End Sub
'================================================================================
Public Sub SetUnassignedWshsMsg()
m_sTitle = "Unassigned Worksheets"
m_sPrompt = "ALL of your data worksheets must be assigned to a " & _
"worksheet type."
End Sub
'================================================================================
Public Sub SetEmptyWshTypesMsg(ByRef UnassignedTypes() As String)
Dim sUnassTypesTbl As String
Dim sPrmpt1 As String
Dim sPrmpt2 As String
Dim i As Byte
m_sTitle = Chr(34) & "Empty" & Chr(34) & " Worksheet Types"
For i = 0 To UBound(UnassignedTypes)
    sUnassTypesTbl = vbTab & UnassignedTypes(i) & vbCr
Next i
sPrmpt1 = "You have defined the following Worksheet Type(s) but have not " & _
"assigned any worksheets to it/them:" & m_s2Crs
sPrmpt2 = _
    vbCr & "Either assign worksheets to the type(s) or return to the the " & _
Chr(34) & "Add/Remove Worksheet Types" & Chr(34) & " form to remove the " & _
"worksheet types which you are not going to use."
m_sPrompt = sPrmpt1 & sUnassTypesTbl & sPrmpt2
End Sub
'================================================================================
Public Sub SetPickValidWshTypeMsg()
m_sTitle = "Invalid Worksheet Type"
m_sPrompt = "Please pick a valid worksheet type."
End Sub
'================================================================================
Public Sub SetUserStoppedMacroMsg(ByVal ErrLogFileName As String)
m_sTitle = "Code Interrupted"
m_sPrompt = _
        "You have elected to halt macro execution." & m_s2Crs & _
        "To be on the safe side we suggest that you close and reopen this " & _
        "file in order to ensure that the macro works as designed." & m_s2Crs & _
        "NOTE:  If you have halted or are halting macro execution because " & _
        "of problems with the workbook please contact the Tax Technology " & _
        "Department.  Further, if that is the case we ask that you also " & _
        "email us the following file:" & m_s2Crs1Tab & _
        ErrLogFileName
End Sub
'================================================================================
Public Sub SetFatalErrorMsg(ByVal ErrLogFileName As String, _
    ByVal OriginatingErrFileName As String, _
    ByVal OriginatingProcName As String, _
    ByVal OriginatingErrMsg As String)

If Len(OriginatingErrMsg) = 0 Then
    OriginatingErrMsg = "[No error message was generated.]"
End If

m_sTitle = "Serious Error Encountered"
m_sPrompt = _
        "This application has encountered a serious error:" & m_s2Crs1Tab & _
        "Details:" & m_sCr2Tabs & _
        OriginatingErrMsg & m_s2Crs1Tab & _
        "Procedure:" & m_sCr2Tabs & _
        OriginatingProcName & m_s2Crs1Tab & _
```

```
MsgBoxHandler - 10

        "File:" & m_sCr2Tabs & _
        OriginatingErrFileName & m_s2Crs & _
        "Please inform us at the Tax Technology Group of the error, and " & _
        "please email us the following file:" & m_s2Crs1Tab & _
        ErrLogFileName & m_s2Crs & _
        "Lastly, we strongly suggest that you discontinue working with this " & _
        "file until we have had an opportunity to resolve this issue."
End Sub
'================================================================================
Public Sub SetDeletedOrRenamedWshsMsg(ByVal NoInvalidWshs As Integer, _
        ByRef InvalidWshs() As String)
Const sEXCESSIVE_NO_WSHS_MSG As String = _
        "Too many worksheets to list individually!"
Dim sInvalidWshsTbl As String
Dim sPrmpt1 As String
Dim sPrmpt2 As String
Dim i As Byte

m_sTitle = "Data Worksheets Deleted or Renamed"

If NoInvalidWshs > 10 Then
    sInvalidWshsTbl = sEXCESSIVE_NO_WSHS_MSG
Else
    For i = 0 To NoInvalidWshs - 1
        sInvalidWshsTbl = sInvalidWshsTbl & vbTab & InvalidWshs(i) & vbCr
    Next i
End If

sPrmpt1 = _
        "Since you last ran your data worksheet through the CART Staging " & _
        "Layer Conversion Tool you have either renamed or deleted the " & _
        "following worksheets:" & m_s2Crs

sPrmpt2 = _
    vbCr & "Because these worksheet names are no longer valid you are " & _
    "required to work through the following CART SL buttons, in order, " & _
    "before you can import your source data:" & m_s2Crs1Tab & _
    "ID Data Worksheets" & m_sCrTab & _
    "Add/Remove Worksheet Types" & m_sCrTab & _
    "Associate Worksheet Types (if you employ more than one worksheet type)" & _
    m_sCrTab & _
    "Set Workhseet Type Dimensions (should you identify any " & _
    Chr(34) & "new" & Chr(34) & " worksheets." & m_s2Crs & _
    "NOTE:  We have already made the appropriate changes to the tables in " & _
    "the " & Chr(34) & "SLValdtnLists" & Chr(34) & " worksheet."

m_sPrompt = sPrmpt1 & sInvalidWshsTbl & sPrmpt2
End Sub
'================================================================================
Public Sub SetProtectedWshsMsg(ByRef ProtctdWshs As Collection)
Dim sWshsTbl As String
Dim sPr1 As String
Dim sPr2 As String
Dim sPr3 As String
Dim i As Byte

m_sTitle = "Data Worksheet(s) Protected"
For i = 1 To ProtctdWshs.Count
    sWshsTbl = sWshsTbl & m_sCrTab & ProtctdWshs.Item(i).name
Next i

sPr1 = "The following data worksheets are protected:"
sPr2 = "You will not be able to import data from your source workbook " & _
        "until these worksheets have their protection disabled."
sPr3 = "NOTE:" & vbCr & _
        "The CART Staging Layer Conversion Tool does NOT " & _
        "edit any source data in your workbook.  However, in cases where " & _
        "the macro identifies " & Chr(34) & "Bad" & Chr(34) & " data it " & _
        "copies the appropriate worksheet into the CART Staging Layer " & _
        "Conversion Tool.  The macro then highlights, in black, the " & _
        Chr(34) & "Bad" & Chr(34) & " rows, columns, and cells.  " & _
```

```
        "We cannot allow you to import your data because your " & _
        "worksheets" & Chr(39) & " protected status remains after being " & _
        "copied and will prevent the macro from highlighting the " & _
        "appropriate cells."

m_sPrompt = sPr1 & vbCr & sWshsTbl & m_s2Crs & sPr2 & m_s3Crs & sPr3
End Sub
'===============================================================================
Public Sub SetEmptyFileDimenCellMsg(ByVal DimenNm As String, _
        ByVal FileDimenCellAddr As String)
Dim sPr1 As String
Dim sPr2 As String
m_sTitle = "No Value for " & DimenNm
sPr1 = "You have said the following cell contains the value for " & _
        DimenNm & ":"
sPr2 = "However, we cannot import your data until you populate this " & _
        "cell with a value."
m_sPrompt = sPr1 & m_s2Crs1Tab & FileDimenCellAddr & m_s2Crs & sPr2
End Sub
'===============================================================================
Public Sub SetEmptyWshDimenCellMsg( _
        ByVal WshTypeNm As String, _
        ByVal WshNm As String, _
        ByVal DimenNm As String, _
        ByVal WshDimenCellAddr As String)
Dim sPr1 As String
Dim sPr2 As String
Dim sPr3 As String
m_sTitle = "No Value for " & DimenNm
sPr1 = "For the " & WshTypeNm & " worksheet type you said the " & _
        "following cell contains the value for " & DimenNm & ":"
sPr2 = "However, on worksheet " & WshNm & " this cell is empty.  " & _
        "(There may be other, similar empty cells.  Unfortunately, we " & _
        "can only prompt you about these as we come upon them during the " & _
        "import process.)"
sPr3 = "Please populate this cell and then re-click the " & _
        Chr(34) & "Import Source Data" & Chr(34) & " button."
m_sPrompt = _
        sPr1 & m_s2Crs1Tab & WshDimenCellAddr & m_s2Crs & sPr2 & m_s2Crs & sPr3
End Sub



'*************************************
'DEVELOPER MESSAGES
'*************************************
'===============================================================================
Public Sub vSetRunDebugModeMsg()
m_sTitle = m_sDEV_MSGBOX_TTL
m_sPrompt = "Run in Debug Mode?"
End Sub
'===============================================================================
Public Sub vSetStepThruModeMsg()
m_sTitle = m_sDEV_MSGBOX_TTL
m_sPrompt = "Execute in Step-Through mode?"
End Sub
'===============================================================================
Public Sub vSetStepThruWsChngMsg()
m_sTitle = m_sDEV_MSGBOX_TTL
m_sPrompt = "Step through worksheet_change event?"
End Sub
'===============================================================================
Public Sub vSetOmitPrintAreaTitlesMsg()
m_sTitle = m_sDEV_MSGBOX_TTL
m_sPrompt = "OMIT Print_Titles and Print_Area names?"
End Sub
'===============================================================================
Public Sub vSetCodeBehineForButtonStillUnderDevelopmentMsg()
m_sTitle = m_sDEV_MSGBOX_TTL
m_sPrompt = "The code-behind for this button is still under " & _
"development."
```

```
MsgBoxHandler - 12

End Sub
'================================================================================
'================================================================================
'================================================================================
'================================================================================
'================================================================================
```

```
'.....................................................................................
'.....................................................................................
'.......        Developer:              William H. White (consultant)
'.......        With:                   TEKsystems Inc.
'.......                                www.teksystems.com
'.......        For:                    AIG
'.......                                Financial Information Systems
'.......                                1 NY Plaza, 15th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         8/16/2013
'.......        Proj finished:          11/21/2013
'.......        Proj revised:           1/15/2014
'.....................................................................................
'.....................................................................................


'.....................................................................................
'.........................  Class-level COMMENTS  ....................................
'This class is nothing more than a workaround of the limitation that _
VBA does not allow you to modify the Class_Initialize() procedure to _
accept parameters.
'.....................................................................................
'.....................................................................................


'*************************************************************************************
'*************************************************************************************
'*************************  CLASS-LEVEL DECLARATIONS  ************************************
'*************************************************************************************
'*************************************************************************************
Option Explicit
Option Compare Binary '...use cAsE senSITive string comparisons




'*************************************************************************************
'*************************************************************************************
'*******************************  PRIVATE MEMBERS  ***********************************
'*************************************************************************************
'*************************************************************************************


'**************************************
'MODULE/CLASS-WIDE CONSTANTS
'**************************************
'Strings...
Private Const m_sMODULE_NAME As String = "ObjectFactory"



'**************************************
'MODULE/CLASS-WIDE VARIABLES
'**************************************



'============================================================
'======================  EVENTS  ============================
'============================================================
'=============================================================================
Private Sub Class_Initialize()
'[VBA QUIRK:  Errors in Intialize, Activate, and Terminate event handlers _
cannot be handled by procedures which create or destroy instances of the _
class. If any such error-handling IS required, create a separate public _
procedure - e.g., the Initialize procedure, below.]
End Sub
```

```
ObjectFactory - 2

'==========================================================================
Private Sub Class_Terminate()
'[VBA QUIRK:  Errors in Intialize, Activate, and Terminate event handlers _
cannot be handled by procedures which create or destroy instances of the _
class. If any such error-handling IS required, create a separate public _
procedure - e.g., the Initialize procedure, below.]

'Since there's really nothing useful an error-handler can do at this point...
On Error Resume Next
'clean-up code...
End Sub




'**************************************************************************
'**************************************************************************
'*****************************  PUBLIC MEMBERS  ***************************
'**************************************************************************
'**************************************************************************


'============================================================
'==================  (Public) FUNCTIONS  ====================
'============================================================
'==========================================================================
Public Function CreateDimension( _
    ByVal name As String, _
    ByVal ReqEnum As ProjDimenReq, _
    ByVal SLFCol As Byte, _
    ByVal ValListLabelXLName As String, _
    ByRef RanOkRESULT As Boolean) As Dimension
'Error-handling declarations...
Const sSOURCE As String = "CreateDimension()"
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim Dimen As Dimension

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Set Dimen = New Dimension
Dimen.InitiateProperties _
        name:=name, _
        ReqType:=ReqEnum, _
        SLFColNmbr:=SLFCol, _
        ValListLabelXLNm:=ValListLabelXLName, _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
Set CreateDimension = Dimen

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Function

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bFUNCTION_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
```

```
ObjectFactory - 3

                Resume
        Else
                Resume ExitPoint
        End If
End Function
'================================================================================
Public Function CreateWshType( _
    ByVal WshTypeName As String, _
    ByRef RanOkRESULT As Boolean) As WshType
'Error-handling declarations...
Const sSOURCE As String = "CreateWshType()"
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim objWshType As WshType

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Set objWshType = New WshType
objWshType.Initialize _
        WshTypeNm:=WshTypeName, _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set CreateWshType = objWshType

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Function

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bFUNCTION_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Function
'================================================================================
Public Function CreateTypeWshLinkForm( _
    ByVal FormCaption As String, _
    ByRef RanOkRESULT As Boolean) As frmTypeWshAssoc
'Error-handling declarations...
Const sSOURCE As String = "CreateTypeWshLinkForm()"
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim formNew As frmTypeWshAssoc

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Set formNew = New frmTypeWshAssoc
formNew.Caption = FormCaption
Set CreateTypeWshLinkForm = formNew

ExitPoint:
```

```
ObjectFactory - 4

On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Function

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bFUNCTION_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Function
'================================================================================
Public Function CreateWbkMandDimensForm( _
    ByVal FormCaption As String, _
    ByRef RanOkRESULT As Boolean) As frmWbkMandDimens
'Error-handling declarations...
Const sSOURCE As String = "CreateWbkMandDimensForm()"
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim formNew As frmWbkMandDimens

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
'If g_bStepThruMode Then Stop
Set formNew = New frmWbkMandDimens
formNew.Initialize _
        DictCompareMeth:=TextCompare, _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

formNew.Caption = FormCaption

Set CreateWbkMandDimensForm = formNew

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Function

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bFUNCTION_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Function
'================================================================================
Public Function CreateWshsToImportForm( _
    ByVal FormCaption As String, _
```

```
ObjectFactory - 5

    ByRef RanOkRESULT As Boolean) As frmWshsToImport
'Error-handling declarations...
Const sSOURCE As String = "CreateWshsToImportForm()"
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim formNew As frmWshsToImport

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Set formNew = New frmWshsToImport
formNew.Caption = FormCaption
Set CreateWshsToImportForm = formNew

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Function

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bFUNCTION_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Function
'================================================================================
Public Function CreateAddRemWshTypesForm( _
    ByVal VirginWbk As Boolean, _
    ByVal SourceWbkFullNm As String, _
    ByVal SourceWbkNm As String, _
    ByVal FormCaption As String, _
    ByRef RanOkRESULT As Boolean) As frmAddRemWshTypes
'Error-handling declarations...
Const sSOURCE As String = "CreateAddRemWshTypesForm()"
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim formNew As frmAddRemWshTypes

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Set formNew = New frmAddRemWshTypes
formNew.Caption = FormCaption
Set CreateAddRemWshTypesForm = formNew

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Function

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
```

```
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bFUNCTION_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Function
'==========================================================================
Public Function CreateWshTypeDimensForm( _
    ByVal WsType As String, _
    ByVal DictCompMeth As CompareMethod, _
    ByVal FormCaption As String, _
    ByRef RanOkRESULT As Boolean) As frmWshTypeDimens
'Error-handling declarations...
Const sSOURCE As String = "CreateWshTypeDimensForm()"
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim formNew As frmWshTypeDimens

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True
If g_bStepThruMode Then Stop

Set formNew = New frmWshTypeDimens

formNew.Initialize _
        WshTypeNm:=WsType, _
        FormCaption:=FormCaption, _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set CreateWshTypeDimensForm = formNew

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Function

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bFUNCTION_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Function
'==========================================================================
Public Function CreateWshTypePickerForm( _
        ByRef RanOkRESULT As Boolean) As frmWshTypePicker
'Error-handling declarations...
Const sSOURCE As String = "CreateWshTypePickerForm()"
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim formNew As frmWshTypePicker

On Error GoTo ErrorHandler
```

```
ObjectFactory - 7

'Assume success until the procedure fails...
RanOkRESULT = True
If g_bStepThruMode Then Stop

Set formNew = New frmWshTypePicker
formNew.ClearAndRepopulateListBox RanOkRESULT:=bCalledProcedureRanOk
Set CreateWshTypePickerForm = formNew

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Function

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bFUNCTION_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Function
'================================================================================
Public Function CreateWshTypesColl( _
    ByRef RanOkRESULT As Boolean) As WshTypes
'Error-handling declarations...
Const sSOURCE As String = "CreateWshTypesColl()"
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
'To insure that we only create one collection...
Dim NewColl As WshTypes

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Set NewColl = New WshTypes
NewColl.Initialize RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

Set CreateWshTypesColl = NewColl

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Function

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bFUNCTION_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
```

```
ObjectFactory - 8

End Function



'============================================================
'==================  (Public) PROCEDURES  ==================
'============================================================
'============================================================================
Public Sub Initialize(ByRef RanOkRESULT As Boolean)
'[NOTE:  Use this procedure whenever there is start-up code which might _
possibly fail.  Put such code here, which is a procedure which must be _
called expicitly, because, as noted above, errors which might occur in _
the Class_Initialize() method CANNOT be trapped by a calling procedure.]

'Error-handling declarations...
Const sSOURCE As String = "Initialize()"
Const bPROCEDURE_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until an error is encountered...
RanOkRESULT = True

'Call a lower-level sub-routine.  Make sure the SUB has a ByRef boolean _
parameter for error-handling...
'''NonEntryPointProcedure RanOkRESULT:= bCalledProcedureRanOk
'''if not bCalledProcedureRanOk then Err.Raise g_lERR_HANDLED

ExitPoint:
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bPROCEDURE_IS_ENTRY_PT
        If bRetraceErrorMode Then
                Else
                Resume ExitPoint
        End If
End Sub
'============================================================================
Public Sub CreateGlobalFormsHandler( _
        ByVal NextStage As ProjNextStage, _
        ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "CreateGlobalFormsHandler()"
Const bPROCEDURE_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
If Not g_cFormsHandler Is Nothing Then GoTo ExitPoint

Set g_cFormsHandler = New FormsHandler
g_cFormsHandler.Initialize _
        NextStg:=NextStage, _
        RanOkRESULT:=bCalledProcedureRanOk

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub
```

```
ObjectFactory - 9


ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bPROCEDURE_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'============================================================================
Public Sub CreateGlobalDimensionHandler( _
        ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const sSOURCE As String = "CreateGlobalDimensionHandler()"
Const bSUB_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
If Not g_cDimHandler Is Nothing Then GoTo ExitPoint

Set g_cDimHandler = New DimensionHandler
g_cDimHandler.Initialize RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'============================================================================
'============================================================================
'============================================================================
'============================================================================
'============================================================================
'============================================================================
'============================================================================
```

```
'.....................................................................................
'.....................................................................................
'.......        Developer:              William H. White (consultant)
'.......        With:                   TEKsystems Inc.
'.......                                www.teksystems.com
'.......        For:                    AIG
'.......                                Financial Information Systems
'.......                                1 NY Plaza, 15th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         8/16/2013
'.......        Proj finished:          11/21/2013
'.......        Proj revised:           1/15/2014
'.....................................................................................
'.....................................................................................

'.....................................................................................
'.........................  Class-level COMMENTS  ....................................
'This is a wrapper class for dimension/vaule pairs.  I would have used _
simple Dictionary types except that I wanted to add properties to identify: _
1) The type of demension/value pairs (e.g., File, Worksheet, etc.) _
2) Source file, source worksheet, etc.
'.....................................................................................
'.....................................................................................


'*************************************************************************************
'*************************************************************************************
'*************************  CLASS-LEVEL DECLARATIONS  ************************************
'*************************************************************************************
'*************************************************************************************
Option Explicit




'*************************************************************************************
'*************************************************************************************
'*******************************  PRIVATE MEMBERS  ***********************************
'*************************************************************************************
'*************************************************************************************

'**************************************
'MODULE/CLASS-WIDE CONSTANTS
'**************************************
'Strings...
Private Const m_sMODULE_NAME As String = "ValueByDimen"


'**************************************
'MODULE/CLASS-WIDE VARIABLES
'**************************************
'Collections...
Private m_dict As Dictionary

'Dimension Scope...
Private m_enumScope As ProjDimenScope

'Amount Type...
Private m_enumAmtType As ProjAmtType

'Strings...
Private m_sSourceWsh As String

'=========================================================
'======================  EVENTS  =========================
'=========================================================
'=================================================================================
```

```
ValueByDimen - 2

Private Sub Class_Initialize()
Set m_dict = New Dictionary
m_dict.CompareMode = TextCompare
End Sub
'=============================================================================
Private Sub Class_Terminate()
'Since there's really nothing useful an error-handler can do at this point...
On Error Resume Next
Set m_dict = Nothing
End Sub




'*****************************************************************************
'*****************************************************************************
'*****************************  PUBLIC MEMBERS  ******************************
'*****************************************************************************
'*****************************************************************************


'============================================================
'======================  PROPERTIES  ======================
'============================================================
'=============================================================================
'CompareMode (read-only)...
Property Get CompareMode() As CompareMethod
CompareMode = m_dict.CompareMode
End Property
'DimensionScope...
Property Let DimensionScope(value As ProjDimenScope)
m_enumScope = value
End Property
Property Get DimensionScope() As ProjDimenScope
DimensionScope = m_enumScope
End Property
'SourceWsh...
Property Let SourceWsh(value As String)
m_sSourceWsh = value
End Property
Property Get SourceWsh() As String
SourceWsh = m_sSourceWsh
End Property
'AmtType...
Property Let AmountType(value As ProjAmtType)
    m_enumAmtType = value
End Property
Property Get AmountType() As ProjAmtType
    AmountType = m_enumAmtType
End Property



'============================================================
'==================  (Public) FUNCTIONS  ==================
'============================================================
'=============================================================================
Public Function Items() As Dictionary
Set Items = m_dict
End Function
'=============================================================================
Public Function Item(index) As String
Item = m_dict(index)
End Function
'=============================================================================
Public Function Count() As Integer
Count = m_dict.Count
End Function
'=============================================================================
Public Function Exists(Dimen As Dimension) As Boolean
```

```
ValueByDimen - 3

Dim bExists As Boolean
bExists = m_dict.Exists(Dimen)
Exists = bExists
End Function
'=============================================================================
Public Function Keys() As Dimension()
Keys = m_dict.Keys
End Function



'============================================================
'==================  (Public) PROCEDURES  ==================
'============================================================
'=============================================================================
Public Sub Add(Dimen As Dimension, value As String)
'No need for error handling; incompatible types will not even compile...
m_dict.Add Key:=Dimen, Item:=value
End Sub
'=============================================================================
Public Sub Remove(Dimen, ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "Remove()"
'Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

m_dict.Remove Dimen

ExitPoint:
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'=============================================================================
Public Sub RemoveAll()
Set m_dict = New Dictionary
End Sub
'=============================================================================
'=============================================================================
'=============================================================================
'=============================================================================
'=============================================================================
'=============================================================================
```

```
WshType - 1

'.....................................................................................
'.....................................................................................
'.......        Developer:              William H. White (consultant)
'.......        With:                   TEKsystems Inc.
'.......                                www.teksystems.com
'.......        For:                    AIG
'.......                                Financial Information Systems
'.......                                1 NY Plaza, 15th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         9/1/2013
'.......        Proj finished:          11/21/2013
'.......        Proj revised:           1/15/2014
'.....................................................................................
'.....................................................................................

'.....................................................................................
'.........................  Class-level COMMENTS  ....................................
'Commentary...
'.....................................................................................
'.....................................................................................


'*************************************************************************************
'*************************************************************************************
'*************************  CLASS-LEVEL DECLARATIONS  ************************************
'*************************************************************************************
'*************************************************************************************
Option Explicit
Option Compare Binary '...use cAsE senSITive string comparisons
Implements ICloneable



'*************************************************************************************
'*************************************************************************************
'*******************************  PRIVATE MEMBERS  ***********************************
'*************************************************************************************
'*************************************************************************************

'**************************************
'MODULE/CLASS-WIDE CONSTANTS
'**************************************
'Strings...
Private Const m_sMODULE_NAME As String = "WshType"


'**************************************
'MODULE/CLASS-WIDE VARIABLES
'**************************************
'Collections...
Private m_coll As Collection
Private m_collKeys As Collection



'Strings...
Private m_sWshTypeNm As String


'=========================================================
'======================  EVENTS  =========================
'=========================================================
'=====================================================================================
Private Sub Class_Initialize()

End Sub
'=====================================================================================
```

```
WshType - 2

Private Sub Class_Terminate()
'Since there's really nothing useful an error-handler can do at this point...
On Error Resume Next
Set m_coll = Nothing
Set m_collKeys = Nothing
End Sub




'******************************************************************************
'******************************************************************************
'*****************************  PUBLIC MEMBERS  *******************************
'******************************************************************************
'******************************************************************************

'============================================================
'======================  PROPERTIES  ========================
'============================================================
'==========================================================================
'Property Name (read-only)...
Property Get name() As String
name = m_sWshTypeNm
End Property
'Property NewEnum...
Public Property Get NewEnum() As IUnknown
        Set NewEnum = m_coll.[_NewEnum]
End Function

'============================================================
'==================  (Public) FUNCTIONS  ====================
'============================================================
'==========================================================================
Public Function Items() As Collection
Set Items = m_coll
End Function
'==========================================================================
Public Function Item(index) As Worksheet
Set Item = m_coll(index)
End Function
'==========================================================================
Public Function Count() As Integer
Count = m_coll.Count
End Function
'==========================================================================
Public Function ContainsItem(WshName As String) As Boolean
Dim bContains As Boolean

If m_coll.Count > 0 Then
        Dim ws As Worksheet
        Dim i As Integer
        For i = 1 To m_coll.Count
                Set ws = m_coll.Item(i)
                If StrComp(ws.name, WshName) = 0 Then
                        bContains = True
                        Exit For
                End If
        Next i
End If
ContainsItem = bContains
End Function
'==========================================================================
Public Function ICloneable_Clone(ByRef RanOkRESULT As Boolean) As Object
'Error-handling declarations...
Const sSOURCE As String = "IClonable_Clone()"
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim clone As WshType
Dim ws As Worksheet
```

```
WshType - 3


On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Set clone = g_cObjFactory.CreateWshType( _
        WshTypeName:=Me.name, _
        RanOkRESULT:=bCalledProcedureRanOk)

For Each ws In Me.Items
    clone.Add ws
Next ws

Set ICloneable_Clone = clone

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Function

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
            ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bFUNCTION_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Function



'============================================================
'==================  (Public) PROCEDURES  ==================
'============================================================
'============================================================================
Public Sub Add(wsh As Worksheet)
'No need for error handling; incompatible types will not even compile...
m_coll.Add Item:=wsh, Key:=wsh.name
m_collKeys.Add Item:=wsh.name, Key:=wsh.name
End Sub
'============================================================================
Public Sub Remove(index, ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "Remove()"
'Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

m_coll.Remove index
m_collKeys.Remove index

ExitPoint:
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
```

```
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'============================================================================
Public Sub RemoveAll()
Set m_coll = New Collection
Set m_collKeys = New Collection
End Sub
'============================================================================
Public Sub Initialize( _
        ByVal WshTypeNm As String, _
        ByRef RanOkRESULT As Boolean)
'[NOTE:  Use this procedure whenever there is start-up code which might _
possibly fail.  Put such code here, which is a procedure which must be _
called expicitly, because, as noted above, errors which might occur in _
the UserForm_Initialize() method CANNOT be trapped by a calling procedure.]
'Error-handling declarations...
Const sSOURCE As String = "Initialize()"
Const bSUB_IS_ENTRY_PT As Boolean = False
'To insure that only one such type is created...
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until an error is encountered...
RanOkRESULT = True

Set m_coll = New Collection
Set m_collKeys = New Collection
m_sWshTypeNm = WshTypeNm

ExitPoint:
Exit Sub

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'============================================================================
'============================================================================
'============================================================================
'============================================================================
'============================================================================
```

```
'.....................................................................................
'.....................................................................................
'.......        Developer:              William H. White (consultant)
'.......        With:                   TEKsystems Inc.
'.......                                www.teksystems.com
'.......        For:                    AIG
'.......                                Financial Information Systems
'.......                                1 NY Plaza, 15th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         9/14/2013
'.......        Proj finished:          11/21/2013
'.......        Proj revised:           1/15/2014
'.....................................................................................
'.....................................................................................

'.....................................................................................
'.........................  Class-level COMMENTS  ....................................
'Commentary...
'.....................................................................................
'.....................................................................................


'*************************************************************************************
'*************************************************************************************
'*************************  CLASS-LEVEL DECLARATIONS  ************************************
'*************************************************************************************
'*************************************************************************************
Option Explicit
Option Compare Binary '...use cAsE senSITive string comparisons



'*************************************************************************************
'*************************************************************************************
'*******************************  PRIVATE MEMBERS  ***********************************
'*************************************************************************************
'*************************************************************************************

'**************************************
'MODULE/CLASS-WIDE CONSTANTS
'**************************************
'Strings...
Private Const m_sMODULE_NAME As String = "WshTypeDimensForms"


'**************************************
'MODULE/CLASS-WIDE VARIABLES
'**************************************
'Collections...
Private m_coll As Collection
Private m_collKeys As Collection



'============================================================
'======================  EVENTS  ============================
'============================================================
'=============================================================================
Private Sub Class_Initialize()
Set m_coll = New Collection
Set m_collKeys = New Collection
End Sub
'=============================================================================
Private Sub Class_Terminate()
'Since there's really nothing useful an error-handler can do at this point...
On Error Resume Next
Set m_coll = Nothing
```

```
WshTypeDimensForms - 2

Set m_collKeys = Nothing
End Sub
'=========================================================
'==================  (Private) FUNCTIONS  ==================
'=========================================================
'=============================================================================
Private Function GetNumericIndexFromStringKey(ByVal Key) As Byte
Dim iKey As Byte
Dim i As Byte
'Get the integer key by looping through the keys collection...
For i = 1 To m_collKeys.Count
    If StrComp(Trim(Key), Trim(m_collKeys(i))) = 0 Then
        iKey = i
        i = m_collKeys.Count
    End If
Next i
GetNumericIndexFromStringKey = iKey
End Function



'*****************************************************************************
'*****************************************************************************
'*****************************  PUBLIC MEMBERS  ******************************
'*****************************************************************************
'*****************************************************************************

'=========================================================
'======================  PROPERTIES  =====================
'=========================================================
'=============================================================================
'Property NewEnum...
Public Property Get NewEnum() As IUnknown
        Set NewEnum = m_coll.[_NewEnum]
End Function

'=========================================================
'==================  (Public) FUNCTIONS  ==================
'=========================================================
'=============================================================================
Public Function Items() As Collection
Set Items = m_coll
End Function
'=============================================================================
Public Function Keys() As Collection
Set Keys = m_collKeys
End Function
'=============================================================================
Public Function Item(index) As frmWshTypeDimens
If IsNumeric(index) Then
    index = Val(index) '...just in case it is passed in as a string
    Set Item = m_coll(index)
Else
    Set Item = m_coll(GetNumericIndexFromStringKey(index))
End If
End Function
'=============================================================================
Public Function Count() As Integer
Count = m_coll.Count
End Function
'=============================================================================
Public Function ContainsItem(WshTypeName As String) As Boolean
Dim bContains As Boolean

If m_coll.Count > 0 Then
        Dim frmWTD As frmWshTypeDimens
        Dim i As Integer
        For i = 1 To m_coll.Count
                Set frmWTD = m_coll.Item(i)
                If StrComp(frmWTD.WshTypeNm, Trim(WshTypeName), _
                        vbTextCompare) = 0 Then
```

```
WshTypeDimensForms - 3

                    bContains = True
                    i = m_coll.Count '...break loop
                End If
        Next i
End If
ContainsItem = bContains
End Function




'=============================================================
'==================  (Public) PROCEDURES  ==================
'=============================================================
'=============================================================================
Public Sub Add(VWshTypeDimensForm As frmWshTypeDimens)
'No need for error handling; incompatible types will not even compile...
    m_coll.Add Item:=VWshTypeDimensForm, Key:=VWshTypeDimensForm.WshTypeNm
    m_collKeys.Add _
            Item:=VWshTypeDimensForm.WshTypeNm, _
            Key:=VWshTypeDimensForm.WshTypeNm
End Sub
'=============================================================================
Public Sub Remove(index, ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "Remove()"
'Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

m_coll.Remove index
m_collKeys.Remove index

ExitPoint:
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'=============================================================================
Public Sub RemoveAll()
Set m_coll = New Collection
Set m_collKeys = New Collection
End Sub
'=============================================================================
'=============================================================================
'=============================================================================
'=============================================================================
```

```
'.....................................................................................
'.....................................................................................
'.......        Developer:              William H. White (consultant)
'.......        With:                   TEKsystems Inc.
'.......                                www.teksystems.com
'.......        For:                    AIG
'.......                                Financial Information Systems
'.......                                1 NY Plaza, 15th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         8/8/2013
'.......        Proj finished:          11/21/2013
'.......        Proj revised:           1/15/2014
'.....................................................................................
'.....................................................................................

'.....................................................................................
'.........................  Class-level COMMENTS  ....................................
'Commentary...
'.....................................................................................
'.....................................................................................


'*************************************************************************************
'*************************************************************************************
'*************************  CLASS-LEVEL DECLARATIONS  ************************************
'*************************************************************************************
'*************************************************************************************
Option Explicit
Option Compare Binary '...use cAsE senSITive string comparisons
Implements ICloneable


'*************************************************************************************
'*************************************************************************************
'*******************************  PRIVATE MEMBERS  ***********************************
'*************************************************************************************
'*************************************************************************************

'**************************************
'MODULE/CLASS-WIDE CONSTANTS
'**************************************
'Strings...
Private Const m_sMODULE_NAME As String = "WshTypes"


'**************************************
'MODULE/CLASS-WIDE VARIABLES
'**************************************
'Collections...
Private m_coll As Collection
Private m_collKeys As Collection


'============================================================
'======================  EVENTS  ============================
'============================================================
'=====================================================================================
Private Sub Class_Initialize()
'Everything is in our Initialize sub (below)...
End Sub
'=====================================================================================
Private Sub Class_Terminate()
'Since there's really nothing useful an error-handler can do at this point...
On Error Resume Next
Set m_coll = Nothing
Set m_collKeys = Nothing
End Sub
```

WshTypes - 2

```
'============================================================
'==================  (Private) FUNCTIONS  ===================
'============================================================
'=============================================================================
Private Function GetNumericIndexFromStringKey(ByVal Key) As Byte
Dim iKey As Byte
Dim i As Byte
'Get the integer key by looping through the keys collection...
For i = 1 To m_collKeys.Count
    If StrComp(Trim(Key), Trim(m_collKeys(i))) = 0 Then
        iKey = i
        i = m_collKeys.Count
    End If
Next i
GetNumericIndexFromStringKey = iKey
End Function




'*****************************************************************************
'*****************************************************************************
'*****************************  PUBLIC MEMBERS  ******************************
'*****************************************************************************
'*****************************************************************************

'============================================================
'======================  PROPERTIES  ========================
'============================================================
'=============================================================================
'Property NewEnum...
Public Property Get NewEnum() As IUnknown
        Set NewEnum = m_coll.[_NewEnum]
End Function

'============================================================
'==================  (Public) FUNCTIONS  ====================
'============================================================
'=============================================================================
Public Function Items() As Collection
Set Items = m_coll
End Function
'=============================================================================
Public Function Keys() As Collection
Set Keys = m_collKeys
End Function
'=============================================================================
Public Function Item(index) As WshType
If IsNumeric(index) Then
    index = Val(index) '...just in case it is passed in as a string
    Set Item = m_coll(index)
Else
    Set Item = m_coll(GetNumericIndexFromStringKey(index))
End If
End Function
'=============================================================================
Public Function Count() As Integer
Count = m_coll.Count
End Function
'=============================================================================
Public Function ContainsItem(ItemName As String) As Boolean
Dim bContains As Boolean

If m_coll.Count > 0 Then
        Dim objWshT As WshType
        Dim i As Integer
        For i = 1 To m_coll.Count
                Set objWshT = m_coll.Item(i)
                If StrComp(objWshT.name, ItemName) = 0 Then
                        bContains = True
                        Exit For
```

```
WshTypes - 3

                End If
        Next i
End If
ContainsItem = bContains
End Function
'==========================================================================
Public Function ICloneable_Clone(ByRef RanOkRESULT As Boolean) As Object
'Error-handling declarations...
Const sSOURCE As String = "ICloneable_Clone()"
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim clone As WshTypes
Dim wtClone As WshType
Dim objWshType As WshType
Dim yWshTypes As Byte
Dim i As Byte

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'CODE HERE...
Set clone = g_cObjFactory.CreateWshTypesColl(RanOkRESULT:=bCalledProcedureRanOk)

For Each objWshType In Me.Items
    Set wtClone = objWshType.ICloneable_Clone(bCalledProcedureRanOk)
        If Not bCalledProcedureRanOk Then Err.Raise g_lERR_HANDLED
    clone.Add wtClone
Next objWshType

Set ICloneable_Clone = clone

ExitPoint:
On Error Resume Next
'Any must-do and/or clean-up code goes here...
Exit Function

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
            ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
            IsEntryPoint:=bFUNCTION_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Function



'============================================================
'==================  (Public) PROCEDURES  ==================
'============================================================
'==========================================================================
Public Sub Add(VWshType As WshType)
'No need for error handling; incompatible types will not even compile...
m_coll.Add Item:=VWshType, Key:=VWshType.name
m_collKeys.Add Item:=VWshType.name, Key:=VWshType.name
End Sub
'==========================================================================
Public Sub Remove(index, ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "Remove()"
```

```
'Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until the procedure fails...
RanOkRESULT = True

'If g_bStepThruMode Then Stop
m_coll.Remove index
m_collKeys.Remove index

ExitPoint:
Exit Sub

ErrorHandler:
        RanOkRESULT = False
        Dim bRetraceErrorMode As Boolean
        ErrorHandling.CentralErrorHandler _
                ModuleName:=m_sMODULE_NAME, _
                ProcedureName:=sSOURCE, _
                StepThroughErrorModeRESULT:=bRetraceErrorMode, _
                IsEntryPoint:=bSUB_IS_ENTRY_PT
        If bRetraceErrorMode Then
                If g_bDebugMode Then Stop
                'So we can step through the code which caused the error...
                Resume
        Else
                Resume ExitPoint
        End If
End Sub
'============================================================================
Public Sub RemoveAll()
Set m_coll = New Collection
Set m_collKeys = New Collection
End Sub
'============================================================================
Public Sub Initialize(ByRef RanOkRESULT As Boolean)
'[NOTE:  Use this procedure whenever there is start-up code which might _
possibly fail.  Put such code here, which is a procedure which must be _
called expicitly, because, as noted above, errors which might occur in _
the UserForm_Initialize() method CANNOT be trapped by a calling procedure.]
'Error-handling declarations...
Const sSOURCE As String = "Initialize()"
Const bSUB_IS_ENTRY_PT As Boolean = False
'To insure that only one such type is created...
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...

On Error GoTo ErrorHandler
'Assume success until an error is encountered...
RanOkRESULT = True

Set m_coll = New Collection
Set m_collKeys = New Collection

ExitPoint:
Exit Sub

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        IsEntryPoint:=bSUB_IS_ENTRY_PT
    If bRetraceErrorMode Then
        If g_bDebugMode Then Stop
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
```

```
    End If
End Sub
'=============================================================================
'=============================================================================
'=============================================================================
'=============================================================================
'=============================================================================
'=============================================================================
```