ThisWorkbook - 1

```
'.....................................................................................
'.....................................................................................
'.......        Module created:         unknown
'.......        Proj finished:          March 21, 2012
'.....................................................................................
'.....................................................................................


'*************************************************************************************
'*************************************************************************************
'*************************  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...

'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)...
Application.EnableCancelKey = xlErrorHandler

On Error GoTo ErrorHandler

'Instantiate some globals variables...
Set g_cMsgBoxHandler = New MsgBoxHandler

'Since this procedure is freqently run when debugging, and therefore _
the boolean might already be set to true...
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
    Else
        'Since this procedure is freqently run when debugging, and therefore _
        the booleans might well NOT be at their default False values...
        g_bDebugMode = False
        g_bStepThruMode = False
    End If
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
    ''''Prompt for stepping through worksheet_change event(s)...
    '''Dim ansStepThruWsChng As VbMsgBoxResult
    '''g_cMsgBoxHandler.vSetStepThruWsChngMsg
    '''If ansStepThruWsChng = vbYes Then g_bStepThruWsChangeEventMode = True
End If

WshDashboard.Activate

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
'================================================================================
Private Sub Workbook_BeforeClose(Cancel As Boolean)
Dim ansSeeLogFiles As VbMsgBoxResult

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

ansSeeLogFiles = vbNo '...default
If g_bSIWbProbs Then
        g_cMsgBoxHandler.SetThereWereProblemFilesMsg
        ansSeeLogFiles = g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbYesNo)
Else
    GoTo ExitPoint
End If

If ansSeeLogFiles = vbNo Then GoTo ExitPoint

'If here we had problems and the user wants to see log files...
Dim sLogFilesPath As String
Dim sLogFileFullNm As String
'Build 1st variable...
sLogFilesPath = ThisWorkbook.Path
If StrComp(Right(sLogFilesPath, 1), Chr(92)) <> 0 Then _
    sLogFilesPath = sLogFilesPath & Chr(92)

'Now open the log files in Notepad...
If g_bSIWbProbs Then
    sLogFileFullNm = sLogFilesPath & g_sSI_PROBLEM_WBKS_LOG
    Shell "Notepad.exe " & sLogFileFullNm, vbNormalFocus
    If Err.Number <> 0 Then
        g_cMsgBoxHandler.SetCannotOpenProbFilesLogMsg sLogFileFullNm
        g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbCritical
        Err.Clear '...for next attempt
    End If
End If

ExitPoint:
'Good-bye!!!
End Sub
```

ThisWorkbook - 3

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

```
WshBrianSummFed - 1

'.....................................................................................
'.....................................................................................
'.......        Module created:         unknown
'.......        Proj finished:          March 21, 2012
'.....................................................................................
'.....................................................................................



'*************************************************************************************
'*************************************************************************************
'*************************  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 = "WshBrianSummFed"




'=============================================================
'======================  EVENTS  ============================
'=============================================================
'=====================================================================================
Private Sub Worksheet_Activate()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "Worksheet_Activate()"
'Operating code declarations...
Dim pt As PivotTable

On Error GoTo ErrorHandler
If WshWkgAreaFed.Range("tbl_Pivot_Fed").Rows.Count > 1 Then
    For Each pt In Me.PivotTables
        pt.RefreshTable
    Next pt
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
```

'==========
'==========
'==========

```
'.....................................................................................
'.....................................................................................
'.......        Module created:         unknown
'.......        Proj finished:          March 21, 2012
'.....................................................................................
'.....................................................................................



'*************************************************************************************
'*************************************************************************************
'*************************  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 = "WshBrianSummSt"




'=============================================================
'======================  EVENTS  ============================
'=============================================================
'=====================================================================================
Private Sub Worksheet_Activate()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "Worksheet_Activate()"
'Operating code declarations...
Dim pt As PivotTable

On Error GoTo ErrorHandler
If WshWkgAreaFed.Range("tbl_Pivot_Fed").Rows.Count > 1 Then
    For Each pt In Me.PivotTables
        pt.RefreshTable
    Next pt
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
```

WshBrianSummSt - 2

'==========
'==========
'==========

```
'.....................................................................................
'.....................................................................................
'.......        Module created:         unknown
'.......        Proj finished:          March 21, 2012
'.....................................................................................
'.....................................................................................


'*************************************************************************************
'*************************************************************************************
'*************************  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 = "wshdashboard"




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

'See comments in Workbook_Open() procedure re:  EnableCancelKey...
Application.EnableCancelKey = xlErrorHandler

On Error GoTo ErrorHandler
If g_bStepThruMode Then Stop
CmdObjHandlers.ShowFolderDlgPickerAndSetDshbdCell _
    RanOkRESULT:=bCalledProcedureRanOk
If Not bCalledProcedureRanOk Then Err.Raise g_lHANDLED_ERROR

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
```

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

'See comments in Workbook_Open() procedure re:  EnableCancelKey...
Application.EnableCancelKey = xlErrorHandler

On Error GoTo ErrorHandler
If g_bStepThruMode Then Stop
If Not g_bDebugMode Then Application.ScreenUpdating = False

CmdObjHandlers.CreateListSIFoldersOnDashbd RanOkRESULT:=bCalledProcedureRanOk
If Not bCalledProcedureRanOk Then Err.Raise g_lHANDLED_ERROR

ExitPoint:
On Error Resume Next
'Must-run procedure/function clean-up code...
WshDashboard.Range("hdrSINos_Dshbd").Select
Application.ScreenUpdating = True
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 btnInitiateQtr_Click()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "btnInitiateQtr_Click()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim ansInitiateQtr As VbMsgBoxResult

'See comments in Workbook_Open() procedure re:  EnableCancelKey...
Application.EnableCancelKey = xlErrorHandler

Application.StatusBar = "Initiating Qtr. Please wait..."
On Error GoTo ErrorHandler
If g_bStepThruMode Then Stop
'Prompt user with an are-you-sure message box...
g_cMsgBoxHandler.SetInitiatingQtrMsg
ansInitiateQtr = g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbYesNo)
If ansInitiateQtr = vbNo Then GoTo ExitPoint
'If here then user said "Yes"...
CmdObjHandlers.InitiateQtrMaestro RanOkRESULT:=bCalledProcedureRanOk
If Not bCalledProcedureRanOk Then Err.Raise g_lHANDLED_ERROR

ExitPoint:
On Error Resume Next
'Must-run procedure/function clean-up code...
Application.StatusBar = vbNullString
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 btnRetrieveDataForQtr_Click()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "btnRetrieveDataForQtr_Click()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim ansRetrieveQtlyData As VbMsgBoxResult
Dim ansSIListHasBeenRun As VbMsgBoxResult

'See comments in Workbook_Open() procedure re:  EnableCancelKey...
Application.EnableCancelKey = xlErrorHandler

On Error GoTo ErrorHandler
If g_bStepThruMode Then Stop

'Prompt user with two are-you-sure message boxes...
g_cMsgBoxHandler.SetIsQtrSetAndSIListCurrentMsg
ansSIListHasBeenRun = g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbYesNoCancel)
Select Case ansSIListHasBeenRun
    Case vbNo, vbCancel
        GoTo ExitPoint
    Case vbYes
        'Do nothing - just continue with code...
End Select

CmdObjHandlers.RetrieveDiscrDataForQtrMaestro RanOkRESULT:=bCalledProcedureRanOk
If Not bCalledProcedureRanOk Then Err.Raise g_lHANDLED_ERROR

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

```
'.....................................................................................
'.....................................................................................
'.......        Module created:         unknown
'.......        Proj finished:          March 21, 2012
'.....................................................................................
'.....................................................................................



'*************************************************************************************
'*************************************************************************************
'*************************  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 = "WshDiscrBySI"




'=============================================================
'======================  EVENTS  ============================
'=============================================================
'=====================================================================================
Private Sub Worksheet_Activate()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "Worksheet_Activate()"
'Operating code declarations...
Dim pt As PivotTable

On Error GoTo ErrorHandler
If WshWkgAreaFed.Range("tbl_Pivot_Fed").Rows.Count > 1 Then
    For Each pt In Me.PivotTables
        pt.RefreshTable
    Next pt
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
'=====================================================================================
```

WshDiscrBySI - 2

'==========
'==========

WshDiscrCatRptFed - 1

```
'.....................................................................................
'.....................................................................................
'.......        Module created:         unknown
'.......        Proj finished:          March 21, 2012
'.....................................................................................
'.....................................................................................


'*************************************************************************************
'*************************************************************************************
'*************************  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 = "WshDiscrCatRptFed"




'=============================================================
'======================  EVENTS  ============================
'=============================================================
'=====================================================================================
Private Sub Worksheet_Activate()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "Worksheet_Activate()"
'Operating code declarations...
Dim pt As PivotTable

On Error GoTo ErrorHandler
If WshWkgAreaFed.Range("tbl_Pivot_Fed").Rows.Count > 1 Then
    For Each pt In Me.PivotTables
        pt.RefreshTable
    Next pt
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
'=====================================================================================
```

WshDiscrCatRptFed - 2

'==========
'==========

```
WshDiscrCatRptSt - 1

'.....................................................................................
'.....................................................................................
'.......        Module created:         unknown
'.......        Proj finished:          March 21, 2012
'.....................................................................................
'.....................................................................................



'*************************************************************************************
'*************************************************************************************
'*************************  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 = "WshDiscrCatRptSt"




'=============================================================
'======================  EVENTS  ===========================
'=============================================================
'=============================================================================================
Private Sub Worksheet_Activate()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "Worksheet_Activate()"
'Operating code declarations...
Dim pt As PivotTable

On Error GoTo ErrorHandler
If WshWkgAreaFed.Range("tbl_Pivot_Fed").Rows.Count > 1 Then
    For Each pt In Me.PivotTables
        pt.RefreshTable
    Next pt
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
'=============================================================================
```

```
WshDiscrCatRptSt - 2

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

WshDOSummFed - 1

```
'.....................................................................................
'.....................................................................................
'.......        Module created:         unknown
'.......        Proj finished:          March 21, 2012
'.....................................................................................
'.....................................................................................



'*************************************************************************************
'*************************************************************************************
'*************************  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 = "WshDOSummFed"




'=============================================================
'======================  EVENTS  ============================
'=============================================================
'=====================================================================================
Private Sub Worksheet_Activate()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "Worksheet_Activate()"
'Operating code declarations...
Dim pt As PivotTable

On Error GoTo ErrorHandler
If WshWkgAreaFed.Range("tbl_Pivot_Fed").Rows.Count > 1 Then
    For Each pt In Me.PivotTables
        pt.RefreshTable
    Next pt
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
'=====================================================================================
```

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

```
'.....................................................................................
'.....................................................................................
'.......        Module created:         unknown
'.......        Proj finished:          March 21, 2012
'.....................................................................................
'.....................................................................................



'*************************************************************************************
'*************************************************************************************
'*************************  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 = "WshDOSummSt"




'=============================================================
'======================  EVENTS  ============================
'=============================================================
'=====================================================================================
Private Sub Worksheet_Activate()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "Worksheet_Activate()"
'Operating code declarations...
Dim pt As PivotTable

On Error GoTo ErrorHandler
If WshWkgAreaFed.Range("tbl_Pivot_Fed").Rows.Count > 1 Then
    For Each pt In Me.PivotTables
        pt.RefreshTable
    Next pt
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
'=====================================================================================
```

```
WshDOSummSt - 2

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

```
'.....................................................................................
'.....................................................................................
'.......        Module created:         unknown
'.......        Proj finished:          March 21, 2012
'.....................................................................................
'.....................................................................................



'*************************************************************************************
'*************************************************************************************
'*************************  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 = "WshETRFed"




'=============================================================
'======================  EVENTS  ============================
'=============================================================
'=====================================================================================
Private Sub Worksheet_Activate()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "Worksheet_Activate()"
'Operating code declarations...
Dim pt As PivotTable

On Error GoTo ErrorHandler
If WshWkgAreaFed.Range("tbl_Pivot_Fed").Rows.Count > 1 Then
    For Each pt In Me.PivotTables
        pt.RefreshTable
    Next pt
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
'=====================================================================================
```

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

WshETRFgnDtlFed - 1

```
'.....................................................................................
'.....................................................................................
'.......        Module created:         unknown
'.......        Proj finished:          March 21, 2012
'.....................................................................................
'.....................................................................................



'*************************************************************************************
'*************************************************************************************
'*************************  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 = "WshETRFgnDtlFed"




'=============================================================
'======================  EVENTS  ============================
'=============================================================
'=====================================================================================
Private Sub Worksheet_Activate()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "Worksheet_Activate()"
'Operating code declarations...
Dim pt As PivotTable

On Error GoTo ErrorHandler
If WshWkgAreaFed.Range("tbl_Pivot_Fed").Rows.Count > 1 Then
    For Each pt In Me.PivotTables
        pt.RefreshTable
    Next pt
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
'=====================================================================================
```

```
WshETRFgnDtlFed - 2

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

```
'.....................................................................................
'.....................................................................................
'.......        Module created:         unknown
'.......        Proj finished:          March 21, 2012
'.....................................................................................
'.....................................................................................



'*************************************************************************************
'*************************************************************************************
'*************************  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 = "WshETRFgnDtlSt"




'=============================================================
'======================  EVENTS  ============================
'=============================================================
'=====================================================================================
Private Sub Worksheet_Activate()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "Worksheet_Activate()"
'Operating code declarations...
Dim pt As PivotTable

On Error GoTo ErrorHandler
If WshWkgAreaFed.Range("tbl_Pivot_Fed").Rows.Count > 1 Then
    For Each pt In Me.PivotTables
        pt.RefreshTable
    Next pt
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
'=====================================================================================
```

WshETRFgnDtlSt - 2

'==========
'==========

```
'.....................................................................................
'.....................................................................................
'.......        Module created:         unknown
'.......        Proj finished:          March 21, 2012
'.....................................................................................
'.....................................................................................



'*************************************************************************************
'*************************************************************************************
'*************************  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 = "WshETRSt"




'=============================================================
'======================  EVENTS  ============================
'=============================================================
'=====================================================================================
Private Sub Worksheet_Activate()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "Worksheet_Activate()"
'Operating code declarations...
Dim pt As PivotTable

On Error GoTo ErrorHandler
If WshWkgAreaFed.Range("tbl_Pivot_Fed").Rows.Count > 1 Then
    For Each pt In Me.PivotTables
        pt.RefreshTable
    Next pt
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
'=============================================================================
```

WshETRSt - 2

'==
'==

WshHiLevelFed - 1

```
'.....................................................................................
'.....................................................................................
'.......        Module created:         unknown
'.......        Proj finished:          March 21, 2012
'.....................................................................................
'.....................................................................................



'*************************************************************************************
'*************************************************************************************
'*************************  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 = "WshHiLevelFed"




'=============================================================
'======================  EVENTS  ===========================
'=============================================================
'=====================================================================================
Private Sub Worksheet_Activate()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "Worksheet_Activate()"
'Operating code declarations...
Dim pt As PivotTable

On Error GoTo ErrorHandler
If WshWkgAreaFed.Range("tbl_Pivot_Fed").Rows.Count > 1 Then
    For Each pt In Me.PivotTables
        pt.RefreshTable
    Next pt
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
'=====================================================================================
```

WshHiLevelFed - 2

'==========
'==========

```
WshHiLevelSt - 1

'.....................................................................................
'.....................................................................................
'.......        Module created:         unknown
'.......        Proj finished:          March 21, 2012
'.....................................................................................
'.....................................................................................



'*************************************************************************************
'*************************************************************************************
'*************************  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 = "WshHiLevelSt"




'=============================================================
'======================  EVENTS  ============================
'=============================================================
'=====================================================================================
Private Sub Worksheet_Activate()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "Worksheet_Activate()"
'Operating code declarations...
Dim pt As PivotTable

On Error GoTo ErrorHandler
If WshWkgAreaFed.Range("tbl_Pivot_Fed").Rows.Count > 1 Then
    For Each pt In Me.PivotTables
        pt.RefreshTable
    Next pt
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
'=====================================================================================
```

```
WshHiLevelSt - 2

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

WshSummaryFed - 1

```
'.....................................................................................
'.....................................................................................
'.......        Module created:         unknown
'.......        Proj finished:          March 21, 2012
'.....................................................................................
'.....................................................................................



'*************************************************************************************
'*************************************************************************************
'*************************  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 = "WshSummaryFed"




'=============================================================
'======================  EVENTS  ============================
'=============================================================
'=====================================================================================
Private Sub Worksheet_Activate()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "Worksheet_Activate()"
'Operating code declarations...
Dim pt As PivotTable

On Error GoTo ErrorHandler
If WshWkgAreaFed.Range("tbl_Pivot_Fed").Rows.Count > 1 Then
    For Each pt In Me.PivotTables
        pt.RefreshTable
    Next pt
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
'=====================================================================================
```

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

WshSummarySt - 1

```
'.....................................................................................
'.....................................................................................
'.......        Module created:         unknown
'.......        Proj finished:          March 21, 2012
'.....................................................................................
'.....................................................................................



'*************************************************************************************
'*************************************************************************************
'*************************  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 = "WshSummarySt"




'=============================================================
'======================  EVENTS  ============================
'=============================================================
'=====================================================================================
Private Sub Worksheet_Activate()
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = True
Const sSOURCE As String = "Worksheet_Activate()"
'Operating code declarations...
Dim pt As PivotTable

On Error GoTo ErrorHandler
If WshWkgAreaFed.Range("tbl_Pivot_Fed").Rows.Count > 1 Then
    For Each pt In Me.PivotTables
        pt.RefreshTable
    Next pt
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
'=====================================================================================
```

```
WshSummarySt - 2

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

```
'.....................................................................................
'.....................................................................................
'.......        Module created:         January 18, 2012
'.......        Proj finished:          March 21, 2012
'.....................................................................................
'.....................................................................................


'.....................................................................................
'.........................  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 = "CmdObjHandlers"




'=============================================================
'==================  (Private) FUNCTIONS  ==================
'=============================================================



'=============================================================
'==================  (Private) PROCEDURES  ==================
'=============================================================
'=======================================================================================
Private Sub RemoveFiltersFromSelectWshs(ByVal ForInitiatingQtr As Boolean, _
    ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "RemoveFiltersFromSelectWshs()"
'Operating code declarations...
Const yWSHS_IN_ALL_CASES As Byte = 4
Const yWSHS_INIT_QTR_ONLY As Byte = 2
Dim awsAllCases(0 To yWSHS_IN_ALL_CASES - 1) As Worksheet
Dim i As Byte

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

Set awsAllCases(0) = WshWkgAreaFed
Set awsAllCases(1) = WshWkgAreaSt
Set awsAllCases(2) = WshCopyFed
Set awsAllCases(3) = WshCopySt

For i = 0 To yWSHS_IN_ALL_CASES - 1
    awsAllCases(i).AutoFilterMode = False
```

```
CmdObjHandlers - 2

Next i

If Not ForInitiatingQtr Then GoTo ExitPoint

Dim awsInitQtrCases(0 To yWSHS_INIT_QTR_ONLY) As Worksheet
Set awsInitQtrCases(0) = WshPrQtrFed
Set awsInitQtrCases(1) = WshPrQtrSt

For i = 0 To yWSHS_INIT_QTR_ONLY - 1
    awsInitQtrCases(i).AutoFilterMode = False
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
'================================================================================
Private Sub CheckStructuralIntegityOfDataTables( _
    ByVal RetrievingDataForQtr As Boolean, _
    ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "CheckStructuralIntegityOfDataTables()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Const yNMBR_DATA_TYPES As Byte = 2
Dim aDataType() As ProjDataType
Dim sXLNmTblHdrs_WkgAr As String
Dim lTtlColsHdrsRngShdHave As Long
Dim bHorizDimensOk As Boolean
Dim i As Byte

'*** NOTE 1:  We only check the dimensions of the named headers regions in this _
procedure because the data regions are all dynamically defined as offsets of _
the named headers regions.
'*** NOTE 2:  This whole procedure COULD be more compact, either by using _
two-dimendional arrays or by using an array of arrays.  However, given the _
limited number of variable we are working with here achieving that compactness _
hardly seems worth the additional layer of complexity we'd be introducing _
into the code.  ~~WHW

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

'OPERATING CODE HERE...
ReDim aDataType(0 To yNMBR_DATA_TYPES - 1)
aDataType(0) = projFederal
aDataType(1) = projState

lTtlColsHdrsRngShdHave = _
    g_yWA_CLASSFCTN_COLS + g_yWA_HDR_DATA_COLS + g_ySIWB_DISCRWS_COLS

'Instantiate variables...
For i = 0 To UBound(aDataType)
    Select Case aDataType(i)
```

```
        Case projFederal
            sXLNmTblHdrs_WkgAr = "hdrsDiscr_WkgArFed"
        Case projState
            sXLNmTblHdrs_WkgAr = "hdrsDiscr_WkgArSt"
    End Select

    'Check dimensions...
    bHorizDimensOk = _
        Utilities.VerifyHorizDimensXLNamedRegion( _
            XLName:=sXLNmTblHdrs_WkgAr, _
            RANGERowToSizeOn:=1, _
            ExpectedNoCols:=lTtlColsHdrsRngShdHave, _
            RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lHANDLED_ERROR
    If Not bHorizDimensOk Then Err.Raise g_lSTRUCTURE_ALTERED
    '***NOTE:  we don't need to check the horizontal dimensions of _
    any other named ranges on the Working Area sheets.  If the full table, _
    the Classifications region or the IDAandData Fields regions are _
    horizontally corrupt then the overall headers range we just checked _
    will have shown up as corrupted.
Next i

If Not RetrievingDataForQtr Then GoTo ExitPoint

Dim sXLNmLkupHdrs_Copy As String
Dim sXLNmLkupHdrs_PrQtr As String
lTtlColsHdrsRngShdHave = g_ySIWB_DISCRWS_COLS + g_yWA_CLASSFCTN_COLS
For i = 0 To UBound(aDataType)
    Select Case aDataType(i)
        Case projFederal
            sXLNmLkupHdrs_Copy = "hdrsDataAndClassCols_CopyFed"
            sXLNmLkupHdrs_PrQtr = "hdrsDataAndClassCols_PrQtrFed"
        Case projState
            sXLNmLkupHdrs_Copy = "hdrsDataAndClassCols_CopySt"
            sXLNmLkupHdrs_PrQtr = "hdrsDataAndClassCols_PrQtrSt"
    End Select

    'Check Copy look-up region...
    bHorizDimensOk = _
        Utilities.VerifyHorizDimensXLNamedRegion( _
            XLName:=sXLNmLkupHdrs_Copy, _
            RANGERowToSizeOn:=1, _
            ExpectedNoCols:=lTtlColsHdrsRngShdHave, _
            RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lHANDLED_ERROR
    If Not bHorizDimensOk Then Err.Raise g_lSTRUCTURE_ALTERED

    'Check Prior Quarter look-up region...
    bHorizDimensOk = _
        Utilities.VerifyHorizDimensXLNamedRegion( _
            XLName:=sXLNmLkupHdrs_PrQtr, _
            RANGERowToSizeOn:=1, _
            ExpectedNoCols:=lTtlColsHdrsRngShdHave, _
            RanOkRESULT:=bCalledProcedureRanOk)
    If Not bCalledProcedureRanOk Then Err.Raise g_lHANDLED_ERROR
    If Not bHorizDimensOk Then Err.Raise g_lSTRUCTURE_ALTERED
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
'============================================================================
Private Sub MoveDataOffWkgArWshs( _
    ByVal InitiatingQtr As Boolean, _
    ByVal WhichData As ProjDataType, _
    ByRef WkgAreaEmptyRESULT As Boolean, _
    ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "MoveDataOffWkgArWshs()"
Const yTARGET_WSHS As Byte = 2
'''Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wsWorkingArea As Worksheet
Dim wsCopy As Worksheet
Dim wsPriorQtr As Worksheet
Dim awshTargets() As Worksheet
Dim rngToClear As Range
Dim rngDest As Range
Dim sXLNmWkgArIDColsAndDiscrData As String
Dim sXLNmWkgArClassifData As String
Dim sXLNmWkgArEntireTable As String
Dim sXLNmWkgArHeadersRow As String
Dim lDestCol As Long
Dim yTgtWshArrayCap As Byte
Dim i As Byte

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

Select Case WhichData
    Case projFederal
        Set wsWorkingArea = WshWkgAreaFed
        Set wsCopy = WshCopyFed
        Set wsPriorQtr = WshPrQtrFed
        sXLNmWkgArIDColsAndDiscrData = "rgnwhdynIDAndDataFlds_WkgArFed"
        sXLNmWkgArClassifData = "rgnwhdynClassFlds_WkgArFed"
        sXLNmWkgArEntireTable = "tblwhdynDiscr_WkgArFed"
        sXLNmWkgArHeadersRow = "hdrsDiscr_WkgArFed"
    Case projState
        Set wsWorkingArea = WshWkgAreaSt
        Set wsCopy = WshCopySt
        Set wsPriorQtr = WshPrQtrSt
        sXLNmWkgArIDColsAndDiscrData = "rgnwhdynIDAndDataFlds_WkgArSt"
        sXLNmWkgArClassifData = "rgnwhdynClassFlds_WkgArSt"
        sXLNmWkgArEntireTable = "tblwhdynDiscr_WkgArSt"
        sXLNmWkgArHeadersRow = "hdrsDiscr_WkgArSt"
End Select

'Bail if Working_Area has no data...
If wsWorkingArea.Range(sXLNmWkgArEntireTable).Rows.Count < 2 Then
    WkgAreaEmptyRESULT = True
    GoTo ExitPoint
End If

ReDim awshTargets(0 To yTARGET_WSHS - 1) As Worksheet
Set awshTargets(0) = wsCopy
Set awshTargets(1) = wsPriorQtr

'We use this variable to control whether or not to include the Prior Qtr _
worksheets in our procedure...
If InitiatingQtr Then yTgtWshArrayCap = 1 '...default = 0
For i = 0 To yTgtWshArrayCap
    awshTargets(i).Activate
    awshTargets(i).UsedRange.ClearContents
Next i
```

```
CmdObjHandlers - 5

'Copy/paste Wkg Area ws discrete data (i.e., sans classification cols)...
wsWorkingArea.Activate
wsWorkingArea.Range(sXLNmWkgArIDColsAndDiscrData).Copy
For i = 0 To yTgtWshArrayCap
    awshTargets(i).Activate
    awshTargets(i).Range("A1").PasteSpecial xlPasteValues
Next i
Application.CutCopyMode = False '...turn off border, clear clipboard

'Copy/paste Wkg Area classification cols (pasting onto RHS of target wsh)...
lDestCol = g_yWA_HDR_DATA_COLS + g_ySIWB_DISCRWS_COLS + 1
wsWorkingArea.Activate
wsWorkingArea.Range(sXLNmWkgArClassifData).Copy
For i = 0 To yTgtWshArrayCap
    awshTargets(i).Activate
    Set rngDest = awshTargets(i).Cells(1, lDestCol)
    rngDest.PasteSpecial xlPasteValues
Next i
Application.CutCopyMode = False '...turn off border, clear clipboard

'Now clear Wkg Area ws, except for headers...
wsWorkingArea.Activate
Set rngToClear = wsWorkingArea.Range(sXLNmWkgArEntireTable) '...prelim
Set rngToClear = _
    rngToClear.Offset(wsWorkingArea.Range(sXLNmWkgArHeadersRow).Rows.Count)
Set rngToClear = rngToClear.Resize(rngToClear.Rows.Count - _
    wsWorkingArea.Range(sXLNmWkgArHeadersRow).Rows.Count)
rngToClear.ClearContents

ExitPoint:
On Error Resume Next
'Must-run procedure/function clean-up code...
Application.CutCopyMode = False '...just to be sure
WshDashboard.Activate
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 SetWkgAreaWkshColHdrs(ByVal WhichData As ProjDataType, _
    ByRef RanOkRESULT As Boolean)

'*******************
'There are several static variables defined in this procedure.  The use of _
static variables eliminates having to repopulate the arrays with completely _
identical values on the second (i.e., STATE) go-round through this procedure...
'*******************

'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "SetWkgAreaWkshColHdrs()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wsWkgAr As Worksheet
Dim i As Byte
Static bArrayAlreadyPopulated As Boolean

'Since there are three "types" of columns I elected to set up three arrays. _
Further, while I almost always use zero-based arrays it seemed appropriate _
```

```
to use a 1-based arrays here (WHW)...

'*********
'As of 1/31/12 the array position of each array element corresponds to the _
column number.  However, if the worksheets or columns are movered around _
THERE IS NO NEED TO MAINTAIN THIS CORRESPONDENCE!!!  (That is, the VBA code _
does not in any way require that, say, the TypColNmNoPairs for column 7 go _
in the 7th spot in its array.)
'*********

Static atypClassfColHdrs(1 To g_yWA_CLASSFCTN_COLS) As TypColNmNoPairs
Static atypDiscrHeaderInfoHds(1 To g_yWA_HDR_DATA_COLS) As TypColNmNoPairs
Static atypSIWbColHdrs(1 To g_ySIWB_DISCRWS_COLS) As TypColNmNoPairs

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

If bArrayAlreadyPopulated Then GoTo ProcessArray

'Populate the first array...
atypClassfColHdrs(1).typColNm = g_sWACLASSFCOL_ETR
atypClassfColHdrs(1).typColNo = g_yWACLASSFCOL_ETR

atypClassfColHdrs(2).typColNm = g_sWACLASSFCOL_ETR_OTHR
atypClassfColHdrs(2).typColNo = g_yWACLASSFCOL_ETR_OTHR

atypClassfColHdrs(3).typColNm = g_sWACLASSFCOL_HI_LVL
atypClassfColHdrs(3).typColNo = g_yWACLASSFCOL_HI_LVL

atypClassfColHdrs(4).typColNm = g_sWACLASSFCOL_SUMM
atypClassfColHdrs(4).typColNo = g_yWACLASSFCOL_SUMM

atypClassfColHdrs(5).typColNm = g_sWACLASSFCOL_SEG
atypClassfColHdrs(5).typColNo = g_yWACLASSFCOL_SEG

atypClassfColHdrs(6).typColNm = g_sWACLASSFCOL_DISC_OP
atypClassfColHdrs(6).typColNo = g_yWACLASSFCOL_DISC_OP


'Populate the first array...
atypDiscrHeaderInfoHds(1).typColNm = g_sWAHDRIDCOL_PER
atypDiscrHeaderInfoHds(1).typColNo = g_yWAHDRIDCOL_PER

atypDiscrHeaderInfoHds(2).typColNm = g_sWAHDRIDCOL_SI
atypDiscrHeaderInfoHds(2).typColNo = g_yWAHDRIDCOL_SI

atypDiscrHeaderInfoHds(3).typColNm = g_sWAHDRIDCOL_SI_DESCR
atypDiscrHeaderInfoHds(3).typColNo = g_yWAHDRIDCOL_SI_DESCR


'Populate the third array...
atypSIWbColHdrs(1).typColNm = g_sSIWSCOL_KEY
atypSIWbColHdrs(1).typColNo = g_ySIWSCOL_KEY

atypSIWbColHdrs(2).typColNm = g_sSIWSCOL_PR_QTR_KEY
atypSIWbColHdrs(2).typColNo = g_ySIWSCOL_PR_QTR_KEY

atypSIWbColHdrs(3).typColNm = g_sSIWSCOL_DISCR_ITM_DESC
atypSIWbColHdrs(3).typColNo = g_ySIWSCOL_DISCR_ITM_DESC

atypSIWbColHdrs(4).typColNm = g_sSIWSCOL_DISCR_ITM_CAT
atypSIWbColHdrs(4).typColNo = g_ySIWSCOL_DISCR_ITM_CAT

atypSIWbColHdrs(5).typColNm = g_sSIWSCOL_TAX_TYPE
atypSIWbColHdrs(5).typColNo = g_ySIWSCOL_TAX_TYPE

atypSIWbColHdrs(6).typColNm = g_sSIWSCOL_ORD_RCG
atypSIWbColHdrs(6).typColNo = g_ySIWSCOL_ORD_RCG

atypSIWbColHdrs(7).typColNm = g_sSIWSCOL_DISCR_TYPE
atypSIWbColHdrs(7).typColNo = g_ySIWSCOL_DISCR_TYPE
```

```
CmdObjHandlers - 7

atypSIWbColHdrs(8).typColNm = g_sSIWSCOL_COMP_NM
atypSIWbColHdrs(8).typColNo = g_ySIWSCOL_COMP_NM

atypSIWbColHdrs(9).typColNm = g_sSIWSCOL_RO_NMBR
atypSIWbColHdrs(9).typColNo = g_ySIWSCOL_RO_NMBR

atypSIWbColHdrs(10).typColNm = g_sSIWSCOL_CNTRY
atypSIWbColHdrs(10).typColNo = g_ySIWSCOL_CNTRY

atypSIWbColHdrs(11).typColNm = g_sSIWSCOL_APB_STATUS
atypSIWbColHdrs(11).typColNo = g_ySIWSCOL_APB_STATUS

atypSIWbColHdrs(12).typColNm = g_sSIWSCOL_RPTG_UNT_US_MEMB_NON
atypSIWbColHdrs(12).typColNo = g_ySIWSCOL_RPTG_UNT_US_MEMB_NON

atypSIWbColHdrs(13).typColNm = g_sSIWSCOL_IN_OUT_PER_Q1
atypSIWbColHdrs(13).typColNo = g_ySIWSCOL_IN_OUT_PER_Q1

atypSIWbColHdrs(14).typColNm = g_sSIWSCOL_IN_OUT_SCOPE_Q1
atypSIWbColHdrs(14).typColNo = g_ySIWSCOL_IN_OUT_SCOPE_Q1

atypSIWbColHdrs(15).typColNm = g_sSIWSCOL_FX_RT_Q1
atypSIWbColHdrs(15).typColNo = g_ySIWSCOL_FX_RT_Q1

atypSIWbColHdrs(16).typColNm = g_sSIWSCOL_US_MEMB_LC_Q1
atypSIWbColHdrs(16).typColNo = g_ySIWSCOL_US_MEMB_LC_Q1

atypSIWbColHdrs(17).typColNm = g_sSIWSCOL_US_MEMB_USD_Q1
atypSIWbColHdrs(17).typColNo = g_ySIWSCOL_US_MEMB_USD_Q1

atypSIWbColHdrs(18).typColNm = g_sSIWSCOL_US_MEMB_FTC_USD_Q1
atypSIWbColHdrs(18).typColNo = g_ySIWSCOL_US_MEMB_FTC_USD_Q1

atypSIWbColHdrs(19).typColNm = g_sSIWSCOL_US_MEMB_NET_FTC_USD_Q1
atypSIWbColHdrs(19).typColNo = g_ySIWSCOL_US_MEMB_NET_FTC_USD_Q1

atypSIWbColHdrs(20).typColNm = g_sSIWSCOL_NONMEMB_LC_Q1
atypSIWbColHdrs(20).typColNo = g_ySIWSCOL_NONMEMB_LC_Q1

atypSIWbColHdrs(21).typColNm = g_sSIWSCOL_NONMEMB_USD_Q1
atypSIWbColHdrs(21).typColNo = g_ySIWSCOL_NONMEMB_USD_Q1

atypSIWbColHdrs(22).typColNm = g_sSIWSCOL_CORP_TX_APPR_REQ_Q1
atypSIWbColHdrs(22).typColNo = g_ySIWSCOL_CORP_TX_APPR_REQ_Q1

atypSIWbColHdrs(23).typColNm = g_sSIWSCOL_ATOI_IMPCT_USD_Q1
atypSIWbColHdrs(23).typColNo = g_ySIWSCOL_ATOI_IMPCT_USD_Q1

atypSIWbColHdrs(24).typColNm = g_sSIWSCOL_US_GAAP_IMPCT_Q1
atypSIWbColHdrs(24).typColNo = g_ySIWSCOL_US_GAAP_IMPCT_Q1

atypSIWbColHdrs(25).typColNm = g_sSIWSCOL_IN_OUT_PER_Q2
atypSIWbColHdrs(25).typColNo = g_ySIWSCOL_IN_OUT_PER_Q2

atypSIWbColHdrs(26).typColNm = g_sSIWSCOL_IN_OUT_SCOPE_Q2
atypSIWbColHdrs(26).typColNo = g_ySIWSCOL_IN_OUT_SCOPE_Q2

atypSIWbColHdrs(27).typColNm = g_sSIWSCOL_FX_RT_Q2
atypSIWbColHdrs(27).typColNo = g_ySIWSCOL_FX_RT_Q2

atypSIWbColHdrs(28).typColNm = g_sSIWSCOL_US_MEMB_LC_Q2
atypSIWbColHdrs(28).typColNo = g_ySIWSCOL_US_MEMB_LC_Q2

atypSIWbColHdrs(29).typColNm = g_sSIWSCOL_US_MEMB_USD_Q2
atypSIWbColHdrs(29).typColNo = g_ySIWSCOL_US_MEMB_USD_Q2

atypSIWbColHdrs(30).typColNm = g_sSIWSCOL_US_MEMB_FTC_USD_Q2
atypSIWbColHdrs(30).typColNo = g_ySIWSCOL_US_MEMB_FTC_USD_Q2

atypSIWbColHdrs(31).typColNm = g_sSIWSCOL_US_MEMB_NET_FTC_USD_Q2
atypSIWbColHdrs(31).typColNo = g_ySIWSCOL_US_MEMB_NET_FTC_USD_Q2

atypSIWbColHdrs(32).typColNm = g_sSIWSCOL_NONMEMB_LC_Q2
```

```
atypSIWbColHdrs(32).typColNo = g_ySIWSCOL_NONMEMB_LC_Q2

atypSIWbColHdrs(33).typColNm = g_sSIWSCOL_NONMEMB_USD_Q2
atypSIWbColHdrs(33).typColNo = g_ySIWSCOL_NONMEMB_USD_Q2

atypSIWbColHdrs(34).typColNm = g_sSIWSCOL_CORP_TX_APPR_REQ_Q2
atypSIWbColHdrs(34).typColNo = g_ySIWSCOL_CORP_TX_APPR_REQ_Q2

atypSIWbColHdrs(35).typColNm = g_sSIWSCOL_ATOI_IMPCT_USD_Q2
atypSIWbColHdrs(35).typColNo = g_ySIWSCOL_ATOI_IMPCT_USD_Q2

atypSIWbColHdrs(36).typColNm = g_sSIWSCOL_US_GAAP_IMPCT_Q2
atypSIWbColHdrs(36).typColNo = g_ySIWSCOL_US_GAAP_IMPCT_Q2

atypSIWbColHdrs(37).typColNm = g_sSIWSCOL_IN_OUT_PER_Q3
atypSIWbColHdrs(37).typColNo = g_ySIWSCOL_IN_OUT_PER_Q3

atypSIWbColHdrs(38).typColNm = g_sSIWSCOL_IN_OUT_SCOPE_Q3
atypSIWbColHdrs(38).typColNo = g_ySIWSCOL_IN_OUT_SCOPE_Q3

atypSIWbColHdrs(39).typColNm = g_sSIWSCOL_FX_RT_Q3
atypSIWbColHdrs(39).typColNo = g_ySIWSCOL_FX_RT_Q3

atypSIWbColHdrs(40).typColNm = g_sSIWSCOL_US_MEMB_LC_Q3
atypSIWbColHdrs(40).typColNo = g_ySIWSCOL_US_MEMB_LC_Q3

atypSIWbColHdrs(41).typColNm = g_sSIWSCOL_US_MEMB_USD_Q3
atypSIWbColHdrs(41).typColNo = g_ySIWSCOL_US_MEMB_USD_Q3

atypSIWbColHdrs(42).typColNm = g_sSIWSCOL_US_MEMB_FTC_USD_Q3
atypSIWbColHdrs(42).typColNo = g_ySIWSCOL_US_MEMB_FTC_USD_Q3

atypSIWbColHdrs(43).typColNm = g_sSIWSCOL_US_MEMB_NET_FTC_USD_Q3
atypSIWbColHdrs(43).typColNo = g_ySIWSCOL_US_MEMB_NET_FTC_USD_Q3

atypSIWbColHdrs(44).typColNm = g_sSIWSCOL_NONMEMB_LC_Q3
atypSIWbColHdrs(44).typColNo = g_ySIWSCOL_NONMEMB_LC_Q3

atypSIWbColHdrs(45).typColNm = g_sSIWSCOL_NONMEMB_USD_Q3
atypSIWbColHdrs(45).typColNo = g_ySIWSCOL_NONMEMB_USD_Q3

atypSIWbColHdrs(46).typColNm = g_sSIWSCOL_CORP_TX_APPR_REQ_Q3
atypSIWbColHdrs(46).typColNo = g_ySIWSCOL_CORP_TX_APPR_REQ_Q3

atypSIWbColHdrs(47).typColNm = g_sSIWSCOL_ATOI_IMPCT_USD_Q3
atypSIWbColHdrs(47).typColNo = g_ySIWSCOL_ATOI_IMPCT_USD_Q3

atypSIWbColHdrs(48).typColNm = g_sSIWSCOL_US_GAAP_IMPCT_Q3
atypSIWbColHdrs(48).typColNo = g_ySIWSCOL_US_GAAP_IMPCT_Q3

atypSIWbColHdrs(49).typColNm = g_sSIWSCOL_IN_OUT_PER_Q4
atypSIWbColHdrs(49).typColNo = g_ySIWSCOL_IN_OUT_PER_Q4

atypSIWbColHdrs(50).typColNm = g_sSIWSCOL_IN_OUT_SCOPE_Q4
atypSIWbColHdrs(50).typColNo = g_ySIWSCOL_IN_OUT_SCOPE_Q4

atypSIWbColHdrs(51).typColNm = g_sSIWSCOL_FX_RT_Q4
atypSIWbColHdrs(51).typColNo = g_ySIWSCOL_FX_RT_Q4

atypSIWbColHdrs(52).typColNm = g_sSIWSCOL_US_MEMB_LC_Q4
atypSIWbColHdrs(52).typColNo = g_ySIWSCOL_US_MEMB_LC_Q4

atypSIWbColHdrs(53).typColNm = g_sSIWSCOL_US_MEMB_USD_Q4
atypSIWbColHdrs(53).typColNo = g_ySIWSCOL_US_MEMB_USD_Q4

atypSIWbColHdrs(54).typColNm = g_sSIWSCOL_US_MEMB_FTC_USD_Q4
atypSIWbColHdrs(54).typColNo = g_ySIWSCOL_US_MEMB_FTC_USD_Q4

atypSIWbColHdrs(55).typColNm = g_sSIWSCOL_US_MEMB_NET_FTC_USD_Q4
atypSIWbColHdrs(55).typColNo = g_ySIWSCOL_US_MEMB_NET_FTC_USD_Q4

atypSIWbColHdrs(56).typColNm = g_sSIWSCOL_NONMEMB_LC_Q4
atypSIWbColHdrs(56).typColNo = g_ySIWSCOL_NONMEMB_LC_Q4
```

```
atypSIWbColHdrs(57).typColNm = g_sSIWSCOL_NONMEMB_USD_Q4
atypSIWbColHdrs(57).typColNo = g_ySIWSCOL_NONMEMB_USD_Q4

atypSIWbColHdrs(58).typColNm = g_sSIWSCOL_CORP_TX_APPR_REQ_Q4
atypSIWbColHdrs(58).typColNo = g_ySIWSCOL_CORP_TX_APPR_REQ_Q4

atypSIWbColHdrs(59).typColNm = g_sSIWSCOL_ATOI_IMPCT_USD_Q4
atypSIWbColHdrs(59).typColNo = g_ySIWSCOL_ATOI_IMPCT_USD_Q4

atypSIWbColHdrs(60).typColNm = g_sSIWSCOL_US_GAAP_IMPCT_Q4
atypSIWbColHdrs(60).typColNo = g_ySIWSCOL_US_GAAP_IMPCT_Q4

atypSIWbColHdrs(61).typColNm = g_sSIWSCOL_US_MEMB_LC_YTD
atypSIWbColHdrs(61).typColNo = g_ySIWSCOL_US_MEMB_LC_YTD

atypSIWbColHdrs(62).typColNm = g_sSIWSCOL_US_MEMB_USD_YTD
atypSIWbColHdrs(62).typColNo = g_ySIWSCOL_US_MEMB_USD_YTD

atypSIWbColHdrs(63).typColNm = g_sSIWSCOL_US_MEMB_FTC_USD_YTD
atypSIWbColHdrs(63).typColNo = g_ySIWSCOL_US_MEMB_FTC_USD_YTD

atypSIWbColHdrs(64).typColNm = g_sSIWSCOL_US_MEMB_NET_FTC_USD_YTD
atypSIWbColHdrs(64).typColNo = g_ySIWSCOL_US_MEMB_NET_FTC_USD_YTD

atypSIWbColHdrs(65).typColNm = g_sSIWSCOL_NONMEMB_LC_YTD
atypSIWbColHdrs(65).typColNo = g_ySIWSCOL_NONMEMB_LC_YTD

atypSIWbColHdrs(66).typColNm = g_sSIWSCOL_NONMEMB_USD_YTD
atypSIWbColHdrs(66).typColNo = g_ySIWSCOL_NONMEMB_USD_YTD

atypSIWbColHdrs(67).typColNm = g_sSIWSCOL_ATOI_IMPCT_USD_YTD
atypSIWbColHdrs(67).typColNo = g_ySIWSCOL_ATOI_IMPCT_USD_YTD

atypSIWbColHdrs(68).typColNm = g_sSIWSCOL_US_GAAP_IMPCT_YTD
atypSIWbColHdrs(68).typColNo = g_ySIWSCOL_US_GAAP_IMPCT_YTD

'Set static variable, so we can retain (and, therefore, skip over) all these _
definitions on our next trip through this procedure...
bArrayAlreadyPopulated = True

ProcessArray:
Select Case WhichData
    Case projFederal
        Set wsWkgAr = WshWkgAreaFed
    Case projState
        Set wsWkgAr = WshWkgAreaSt
End Select

wsWkgAr.Activate

'Plug in the column headers for the classification columns...
For i = 1 To UBound(atypClassfColHdrs)
    wsWkgAr.Cells(1, atypClassfColHdrs(i).typColNo) = _
        atypClassfColHdrs(i).typColNm
Next i

'Plug in the column headers for the Discrete Header info columns (shifting _
over for the Classification columns we have just populated)...
For i = 1 To UBound(atypDiscrHeaderInfoHds)
    wsWkgAr.Cells(1, atypDiscrHeaderInfoHds(i).typColNo) = _
        atypDiscrHeaderInfoHds(i).typColNm
Next i

'Plug in the column headers for the classification columns (shifting _
over for both the Classification and SI Header info columns we have _
just populated)...
For i = 1 To UBound(atypSIWbColHdrs)
    wsWkgAr.Cells(1, atypSIWbColHdrs(i).typColNo + _
        g_yWA_CLASSFCTN_COLS + g_yWA_HDR_DATA_COLS) = _
            atypSIWbColHdrs(i).typColNm
Next i
```

```
CmdObjHandlers - 10


ExitPoint:
On Error Resume Next
'Must-run procedure/function clean-up code...
WshDashboard.Activate
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 UpdateProblemFileLog( _
    ByVal ProblemType As ProjFileProb, _
    ByVal ProbFilePath As String, _
    ByVal ProbFileName As String, _
    Optional ProbWshName As String)
Const sERR_MSG_FINDING1 As String = "The following file could not be found:  "
Const sERR_MSG_FINDING2 As String = "Within this folder:  "
Const sERR_MSG_OPENING As String = "The following file could not be opened:  "
Const sERR_MSG_CORRUPT1 As String = "The following file had corrupt data:  "
Const sERR_MSG_CORRUPT2 As String = "In the following worksheet/tab:  "
Const sERR_MSG_GENL As String = _
    "There were problems processing data within the following file:  "
Dim sLogFileFullNm As String
Dim sLogText As String
Dim sDateTimeStamp As String
Dim iFileNmbr As Integer

'Since we really don't want to allow errors here...
On Error Resume Next

'Build log file name (several steps)...
sLogFileFullNm = ThisWorkbook.Path
If StrComp(Right(sLogFileFullNm, 1), Chr(92)) <> 0 Then _
    sLogFileFullNm = sLogFileFullNm & Chr(92)
sLogFileFullNm = sLogFileFullNm & g_sSI_PROBLEM_WBKS_LOG

'Open the log file, write out the error info, then close the file...
iFileNmbr = FreeFile()
Open sLogFileFullNm For Append As iFileNmbr
Print #iFileNmbr, Format(Now(), "mm/dd/yyyy hh:mm AMPM"); "..."
Select Case ProblemType
    Case projCannotFind
        Print #iFileNmbr, sERR_MSG_FINDING1; ProbFileName
        Print #iFileNmbr, sERR_MSG_FINDING2; ProbFilePath
    Case projCannotOpen
        Print #iFileNmbr, sERR_MSG_OPENING; ProbFilePath; ProbFileName
    Case projSIWbDiscrWsCols
        Print #iFileNmbr, sERR_MSG_CORRUPT1; ProbFilePath; ProbFileName
        Print #iFileNmbr, sERR_MSG_CORRUPT2; ProbWshName
    Case projGenlSIWbProbs
        Print #iFileNmbr, sERR_MSG_GENL; ProbFilePath; ProbFileName
End Select
Close iFileNmbr
End Sub
'============================================================================
Private Sub RetrieveDataFmSIWbk( _
    ByVal SIWkbk As Workbook, _
    ByVal SI As String, _
    ByVal WhichData As ProjDataType, _
```

```
    ByRef WbTblsCorruptRESULT As Boolean, _
    ByRef StopProcessingFilesRESULT As Boolean, _
    ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "RetrieveDataFmSIWbk()"
Dim bCalledProcedureRanOk As Boolean
'''Dim sErrSourceFileFullNm As String
'Operating code declarations...
Dim wshWkgArea As Worksheet
Dim wsDiscreteInSIWb As Worksheet
Dim rngSingleValPasteAreaWkgArWs As Range
Dim rngWkgArDataDest As Range
Dim rngNamed As Range
Dim sSIWbDiscreteWshNm As String
Dim sXLNmDisrTblWkgArWs As String
Dim sXLNmFullTableInclHdrsWkgArea As String
Dim sXLNmClassfFldsInclHdrsWkgArea As String
Dim sXLNmIDAndDataFldsInclHdrsWkgArea As String
Dim asXLNmsToRedefine() As String
Dim lDestColWkgArWshForSIWbkData As Long
Dim lLastRow As Long
Dim lLastRowSIWbDiscrWsDescripCol As Long
Dim lRowsDataFmSIWb As Long
Dim lVertSizeWkgArWs As Long
Dim lNewVertSize As Long
Dim bHorizDimensSIWbOk As Boolean
Dim i As Byte

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

Select Case WhichData
    Case projFederal
        Set wshWkgArea = WshWkgAreaFed
        sSIWbDiscreteWshNm = "DISCRETE"
        sXLNmDisrTblWkgArWs = "tblwhdynDiscr_WkgArFed"
        sXLNmClassfFldsInclHdrsWkgArea = "rgnwhdynClassFlds_WkgArFed"
        sXLNmIDAndDataFldsInclHdrsWkgArea = "rgnwhdynIDAndDataFlds_WkgArFed"
        sXLNmFullTableInclHdrsWkgArea = "tblwhdynDiscr_WkgArFed"
    Case projState
        Set wshWkgArea = WshWkgAreaSt
        sSIWbDiscreteWshNm = "STATE_DISCRETE"
        sXLNmDisrTblWkgArWs = "tblwhdynDiscr_WkgArSt"
        sXLNmClassfFldsInclHdrsWkgArea = "rgnwhdynClassFlds_WkgArSt"
        sXLNmIDAndDataFldsInclHdrsWkgArea = "rgnwhdynIDAndDataFlds_WkgArSt"
        sXLNmFullTableInclHdrsWkgArea = "tblwhdynDiscr_WkgArSt"
End Select

Set wsDiscreteInSIWb = SIWkbk.Worksheets(sSIWbDiscreteWshNm)
'If a worksheet is NOT protected this bit of code is simply ignored...
wsDiscreteInSIWb.Unprotect g_sWSH_PASSWORD

'Verify structural integrity of the Discrete sheet in the SI Wbk...
bHorizDimensSIWbOk = _
    Utilities.VerifyHorizDimensWshRegion( _
        WshName:=sSIWbDiscreteWshNm, _
        FirstWshColOfRegion:=1, _
        WshRowToSizeOn:=g_ySIWB_DISCRWS_HDR_ROW, _
        ExpectedNoCols:=g_ySIWB_DISCRWS_COLS, _
        RanOkRESULT:=bCalledProcedureRanOk, _
        ParentWb:=SIWkbk)
If Not bHorizDimensSIWbOk Then
    'Set problem boolean and log the issue...
    g_bSIWbProbs = True
    UpdateProblemFileLog _
        ProblemType:=projSIWbDiscrWsCols, _
        ProbFilePath:=SIWkbk.Path, _
        ProbFileName:=SIWkbk.Name, _
        ProbWshName:=sSIWbDiscreteWshNm
    'Ask user whether he wants to stop or continue...
    Dim ansContinueOpeningFiles As VbMsgBoxResult
```

```
    g_cMsgBoxHandler.SetSIWbDiscrWshCorruptedMsg _
        FileFullNm:=SIWkbk.FullName, _
        DiscreteWshNm:=sSIWbDiscreteWshNm, _
        LogFileFullNm:=ThisWorkbook.Path & "\" & g_sSI_PROBLEM_WBKS_LOG
    ansContinueOpeningFiles = _
        g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbCritical + vbYesNo)
    'Set 2 output booleans...
    WbTblsCorruptRESULT = True
    If ansContinueOpeningFiles = vbNo Then StopProcessingFilesRESULT = True
     'We are done with THIS file whether or not the user wants to process _
     more files...
    GoTo ExitPoint
End If

'If here then the Discrete worksheet has the correct number of columns...
lDestColWkgArWshForSIWbkData = g_yWA_CLASSFCTN_COLS + g_yWA_HDR_DATA_COLS + 1

'To make sure we get accurate row numbers...
wsDiscreteInSIWb.AutoFilterMode = False

lLastRowSIWbDiscrWsDescripCol = _
    Utilities.FindLastRowInColumn( _
        wsParent:=wsDiscreteInSIWb, _
        WshColNmbr:=g_ySIWSCOL_DISCR_ITM_DESC, _
        RanOkRESULT:=bCalledProcedureRanOk, _
        ParentWkbk:=SIWkbk)
If Not bCalledProcedureRanOk Then Err.Raise g_lHANDLED_ERROR
lRowsDataFmSIWb = lLastRowSIWbDiscrWsDescripCol - g_ySIWB_DISCRWS_HDR_ROW

'Bail if there are no data...
If lRowsDataFmSIWb = 0 Then GoTo ExitPoint

'Copy all data SI Workbook/Discrete Wsh onto the clipboard...
SIWkbk.Activate
With wsDiscreteInSIWb
    .Activate
    .Range(Cells(g_ySIWB_DISCRWS_HDR_ROW + 1, 1), _
        Cells(lLastRowSIWbDiscrWsDescripCol, g_ySIWB_DISCRWS_COLS)).Copy
End With

'Paste this onto the bottom (first empty row) of the Wkg Area wsh, in _
the appropriate column...
ThisWorkbook.Activate
With wshWkgArea
    .Activate
    lVertSizeWkgArWs = .Range(sXLNmFullTableInclHdrsWkgArea).Rows.Count
    Set rngWkgArDataDest = _
        .Range(sXLNmFullTableInclHdrsWkgArea).Offset(lVertSizeWkgArWs, _
            lDestColWkgArWshForSIWbkData - 1)
End With
Set rngWkgArDataDest = rngWkgArDataDest.Resize(1, 1)
rngWkgArDataDest.PasteSpecial xlPasteValues


'*********
'Now handle the header/ID data from the SI wb...
'*********
'Header/ID Item 1:  Period...
'Define the appropriate (empty) cells in the Wkg Area ws (2-steps)...
Set rngSingleValPasteAreaWkgArWs = _
    rngWkgArDataDest.Resize(lRowsDataFmSIWb, 1) '...vertical resizing
Set rngSingleValPasteAreaWkgArWs = _
    rngSingleValPasteAreaWkgArWs.Offset(0, _
        g_yWAHDRIDCOL_PER - rngSingleValPasteAreaWkgArWs.Column)
'"Paste" the header value across the destination cells...
rngSingleValPasteAreaWkgArWs = SIWkbk.Names("PERIOD").RefersToRange

'**NOTE:  For the next 2 items we (re-)define the appropriate cells in the _
Wkg Area ws by simply offsetting from the existing range location...

'Header/ID Item 2:  SI Number...
Set rngSingleValPasteAreaWkgArWs = _
    rngSingleValPasteAreaWkgArWs.Offset(0, _
```

```
        g_yWAHDRIDCOL_SI - g_yWAHDRIDCOL_PER)
'"Paste" the header value across the destination cells...
rngSingleValPasteAreaWkgArWs = SIWkbk.Names("SI_No").RefersToRange

'Header/ID Item 3:  SI Name...
Set rngSingleValPasteAreaWkgArWs = _
    rngSingleValPasteAreaWkgArWs.Offset(0, _
        g_yWAHDRIDCOL_SI_DESCR - g_yWAHDRIDCOL_SI)
'"Paste" the header value across the destination cells...
rngSingleValPasteAreaWkgArWs = SIWkbk.Names("SI_NAME").RefersToRange


ExitPoint:
On Error Resume Next
'Must-run procedure/function clean-up code...
Application.CutCopyMode = False '...turn off border, clear clipboard
Exit Sub

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        FileNm:=SIWkbk.FullName, _
        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 PopulateWkgArWsClassfColsViaVLkups( _
    ByVal WhichData As ProjDataType, _
    ByVal NewRows As Long, _
    ByVal Embed2ndLkUpUsingPrQtrKey As Boolean, _
    ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "PopulateWkgArWsClassfColsViaVLkups()"
'''Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wsWkgArea As Worksheet
Dim wsCopy As Worksheet
Dim wsPrQtr As Worksheet
Dim rngFormulaCells As Range
Dim rngFormulaCell As Range
Dim sXLNmClassifRng As String
Dim sXLNmLkUpRngCopyWs As String
Dim sXLNmLkUpRngPrQtrWs As String
Dim sVLkUpRegKey As String
Dim sVLkUpPrQtrKey As String
Dim sQuotMrksPlusComma As String
Dim yFirstWsColOfLkupRange As Byte
Dim yKeyCol As Byte
Dim yPrQtrKeyCol As Byte
Dim yClassfCol As Byte
Dim yLkUpOffset As Byte
Dim i As Byte

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

Select Case WhichData
    Case projFederal
        Set wsWkgArea = WshWkgAreaFed
        Set wsCopy = WshCopyFed
        Set wsPrQtr = WshPrQtrFed
```

```
        sXLNmClassifRng = "rgnwhdynClassFlds_WkgArFed"
        sXLNmLkUpRngCopyWs = "rgnwhdynLkUp_CopyFed"
        sXLNmLkUpRngPrQtrWs = "rgnwhdynLkUp_PrQtrFed"
    Case projState
        Set wsWkgArea = WshWkgAreaSt
        Set wsCopy = WshCopySt
        Set wsPrQtr = WshPrQtrSt
        sXLNmClassifRng = "rgnwhdynClassFlds_WkgArSt"
        sXLNmLkUpRngCopyWs = "rgnwhdynLkUp_CopySt"
        sXLNmLkUpRngPrQtrWs = "rgnwhdynLkUp_PrQtrSt"
End Select

'When setting our lookup column we will need to adjust for the fact that _
the lookup range does NOT start in column 1...
yFirstWsColOfLkupRange = g_ySIWSCOL_KEY + g_yWA_HDR_DATA_COLS '...4

'NOTE:  We cannot enter formulas directly into the Classification columns _
of the Working area worksheet or we will wipe out the drop-down validations _
which are in in those cells...
Set rngFormulaCells = _
    wsWkgArea.Range(sXLNmClassifRng).Offset(1, _
        g_yWA_CLASSFCTN_COLS + g_yWA_HDR_DATA_COLS + g_ySIWB_DISCRWS_COLS)
Set rngFormulaCells = rngFormulaCells.Resize(1)

'***NOTE:
'chr(34) = quotation mark, i.e. """
'chr(44) = comm, i.e. ","
sQuotMrksPlusComma = Chr(34) & Chr(34) & Chr(44)

'Plug VLookup formulas into the formula cells...
For i = 1 To g_yWA_CLASSFCTN_COLS '...6
    Set rngFormulaCell = rngFormulaCells.Cells(1, i)
    yKeyCol = _
        g_ySIWSCOL_KEY + g_yWA_CLASSFCTN_COLS + g_yWA_HDR_DATA_COLS '...10
    yLkUpOffset = g_ySIWB_DISCRWS_COLS + i
    sVLkUpRegKey = "VLOOKUP(RC" & yKeyCol & "," & sXLNmLkUpRngCopyWs & _
        "," & yLkUpOffset & ",FALSE)"

    If Not Embed2ndLkUpUsingPrQtrKey Then
        rngFormulaCell.FormulaR1C1 = _
            "=IF(ISERROR(" & sVLkUpRegKey & ")," & sQuotMrksPlusComma & _
                "IF(LEN(" & sVLkUpRegKey & ")<1," & sQuotMrksPlusComma & _
                    sVLkUpRegKey & "))"

    Else
        yPrQtrKeyCol = _
            g_ySIWSCOL_PR_QTR_KEY + g_yWA_CLASSFCTN_COLS + _
                g_yWA_HDR_DATA_COLS '...11
        sVLkUpPrQtrKey = _
            "VLOOKUP(RC" & yPrQtrKeyCol & "," & sXLNmLkUpRngPrQtrWs & _
                "," & yLkUpOffset & ",FALSE)"

        rngFormulaCell.FormulaR1C1 = _
            "=IF(ISERROR(" & sVLkUpRegKey & ")," & _
                "IF(ISERROR(" & sVLkUpPrQtrKey & ")," & _
                    sQuotMrksPlusComma & _
                    "IF(LEN(" & sVLkUpPrQtrKey & ")<1," & _
                        sQuotMrksPlusComma & _
                        sVLkUpPrQtrKey & "))," & _
                "IF(LEN(" & sVLkUpRegKey & ")<1," & _
                    "IF(ISERROR(" & sVLkUpPrQtrKey & ")," & _
                        sQuotMrksPlusComma & _
                        "IF(LEN(" & sVLkUpPrQtrKey & ")<1," & _
                            sQuotMrksPlusComma & _
                            sVLkUpPrQtrKey & "))," & _
                    sVLkUpRegKey & "))"
    End If
Next i

'If necessary, copy the vlookup formulas down...
If NewRows > 1 Then
    'Set the paste area (2 steps)...
    Dim rngPasteArea As Range
```

```
    Set rngPasteArea = rngFormulaCells.Offset(1, 0)
    Set rngPasteArea = rngPasteArea.Resize(NewRows - 1)
    'Copy the formulas...
    rngFormulaCells.Copy rngPasteArea
    'Redefine the area containing formulas...
    Set rngFormulaCells = rngFormulaCells.Resize(NewRows)
End If

'Finish up...
rngFormulaCells.Calculate
rngFormulaCells.Copy
wsWkgArea.Range(sXLNmClassifRng).Offset(1, 0).PasteSpecial xlPasteValues
rngFormulaCells.EntireColumn.Delete xlShiftToLeft

ExitPoint:
On Error Resume Next
'Must-run procedure/function clean-up code...
Application.CutCopyMode = False '...turn off border, clear clipboard
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 ShowFolderDlgPickerAndSetDshbdCell(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "ShowFolderDlgPickerAndSetDshbdCell()"
'Operating code declarations...
Dim fdSIs As FileDialog
Dim sCurrentPath As String

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

sCurrentPath = WshDashboard.Range("ptrSIFolders_Dshbd")
WshDashboard.Range("ptrSIFolders_Dshbd").ClearContents

Set fdSIs = Application.FileDialog(msoFileDialogFolderPicker)
With fdSIs
    .AllowMultiSelect = False
    .ButtonName = "Select"
    If Not sCurrentPath = vbNullString Then .InitialFileName = sCurrentPath
    .Title = "Select Parent Folder of SI Sub-Folders"
    '.Show method returns a long.  -1 ==> Selection made.  0 ==> Cxl btn...
```

```
    If .Show = -1 Then _
        WshDashboard.Range("ptrSIFolders_Dshbd") = .SelectedItems(1) & "\"
End With

ExitPoint:
On Error Resume Next
'Must-run procedure/function clean-up code...
Set fdSIs = Nothing
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 CreateListSIFoldersOnDashbd(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "CreateListSIFoldersOnDashbd()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim fso As Scripting.FileSystemObject
Dim fldrSIParent As Scripting.Folder
Dim fldrSI As Scripting.Folder
Dim rngDshbdSIFoldersList As Range
Dim sSIFoldersPath As String
Dim iLastRow As Integer
Dim i As Integer

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

sSIFoldersPath = WshDashboard.Range("ptrSIFolders_Dshbd")
sSIFoldersPath = sSIFoldersPath

'Trap for the user not having picked a path...
If sSIFoldersPath = vbNullString Then
    g_cMsgBoxHandler.SetNoSIFolderMsg
    g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbExclamation
    GoTo ExitPoint
End If

'Trap for user somehow not having picked a valid path...
If Not Utilities.DoesFolderOrFileExist(sSIFoldersPath) Then
    g_cMsgBoxHandler.SetInvalidSIFolderMsg
    g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbExclamation
    GoTo ExitPoint
End If

'If here then we have a valid path for the SI folders...
Utilities.CreateXLNameForSingleColDataRange _
    XLNameToDefine:="coldoSINos_Dshbd", _
    HeaderCell:=WshDashboard.Range("hdrSINos_Dshbd"), _
    FullColumn:=WshDashboard.Range("colSINos_Dshbd"), _
    IncludeHdrCellInDefinedRange:=False, _
    ThrowProjTerminatingErrIfDataRangeEmpty:=False, _
    RanOkRESULT:=bCalledProcedureRanOk
If Not bCalledProcedureRanOk Then Err.Raise g_lHANDLED_ERROR

Set rngDshbdSIFoldersList = WshDashboard.Range("coldoSINos_Dshbd")
```

```
rngDshbdSIFoldersList.ClearContents

'Populate the dashboard with the latest and greatest...
Set fso = New Scripting.FileSystemObject
i = 1
Set fldrSIParent = fso.GetFolder(sSIFoldersPath)
For Each fldrSI In fldrSIParent.SubFolders
    'Use Item instead of Cells property because Item can extend beyond the _
    original range...
    rngDshbdSIFoldersList.Item(i, 1) = fldrSI.Name
    i = i + 1
Next fldrSI

'i will always be 1 more than the total number of subfolders found.  However, _
if user picks a folder without ANY sub-folder we need to adjust i so that we _
at least create a 1-cell (empty) range.  If we don't do this the procedure _
will throw a (trapped, of course!) run-time error...
If i = 1 Then i = 2

'Redefine our range and the XL Name...
Set rngDshbdSIFoldersList = rngDshbdSIFoldersList.Resize(i - 1, 1)
ThisWorkbook.Names.Add Name:="coldoSINos_Dshbd", _
    RefersTo:="=" & WshDashboard.Name & "!" & rngDshbdSIFoldersList.Address

'Sort the list...
rngDshbdSIFoldersList.Sort _
    Key1:=rngDshbdSIFoldersList.Cells(1, 1), _
    Order1:=xlAscending, _
    Header:=xlNo, _
    OrderCustom:=1, _
    MatchCase:=False, _
    Orientation:=xlSortRows, _
    DataOption1:=xlSortNormal

ExitPoint:
On Error Resume Next
'Must-run procedure/function clean-up code...
Application.Calculate
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 InitiateQtrMaestro(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "InitiateQtrMaestro()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Const yNMBR_DATA_TYPES As Byte = 2
Dim aDataType() As ProjDataType
Dim bUnused As Boolean
Dim bNoFedWkgArData As Boolean
Dim bNoStWkgArData As Boolean
Dim i As Byte

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

```
Application.ScreenUpdating = False
'Make sure there are no filters on any of the sheets being accessed in this _
procedure...
RemoveFiltersFromSelectWshs _
    ForInitiatingQtr:=True, _
    RanOkRESULT:=bCalledProcedureRanOk
If Not bCalledProcedureRanOk Then Err.Raise g_lHANDLED_ERROR

'Before proceeding see if the user has added or deleted any columns in our _
data tables.  This procedure checks both the Federal and State worksheets. _
If the procedure finds any changes it throws a project-terminating error...
CheckStructuralIntegityOfDataTables _
    RetrievingDataForQtr:=False, _
    RanOkRESULT:=bCalledProcedureRanOk
If Not bCalledProcedureRanOk Then Err.Raise g_lHANDLED_ERROR

ReDim aDataType(0 To yNMBR_DATA_TYPES - 1)
aDataType(0) = projFederal
aDataType(1) = projState

For i = 0 To UBound(aDataType)
    MoveDataOffWkgArWshs _
        InitiatingQtr:=True, _
        WhichData:=aDataType(i), _
        WkgAreaEmptyRESULT:=bUnused, _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lHANDLED_ERROR
    If bUnused Then
        Select Case aDataType(i)
            Case projFederal
                bNoFedWkgArData = True
            Case projState
                bNoStWkgArData = True
        End Select
    End If
Next i

g_cMsgBoxHandler.SetWkgAreaDiscrDataCopiedMsg _
    NoFedDataToCopy:=bNoFedWkgArData, _
    NoStateDataToCopy:=bNoStWkgArData
g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbInformation
WshDashboard.Activate

ExitPoint:
On Error Resume Next
'Must-run procedure/function clean-up code...
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
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'=============================================================================
Public Sub RetrieveDiscrDataForQtrMaestro(ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "RetrieveDiscrDataForQtrMaestro()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
```

```
Const yNMBR_DATA_TYPES As Byte = 2
Dim aDataType() As ProjDataType
Dim wbSI As Workbook
Dim wsWkgAr As Worksheet
Dim rngSINosDshbdWs As Range
Dim rngNamed As Range
Dim cl As Range
Dim sSIFoldersParentPath As String
Dim sSIFolder As String
Dim sYrQtr As String
Dim sSINmbr As String
Dim sSIWbNm As String
Dim sSIWbFullNm As String
Dim sXLNmTblNmWhDiscrWkgAr As String
Dim lVertSizeWkgArWshInclHdr As Long
Dim lNewRows As Long
Dim ansContOpeningFiles As VbMsgBoxResult
Dim bSIWbkExists As Boolean
Dim bSIWbkOpened As Boolean
Dim bSIWbkStructureCorrupt As Boolean
Dim bStopProcessingFiles As Boolean
Dim bNoWkgArData As Boolean
Dim bFirstDataPullForQtrFed As Boolean
Dim bFirstDataPullForQtrSt As Boolean
Dim bUseDblLkupFormula As Boolean
Dim bUpdatedSomeData As Boolean
Dim i As Byte

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

If Not g_bDebugMode Then Application.ScreenUpdating = False
sSIFoldersParentPath = WshDashboard.Range("ptrSIFolders_Dshbd")
sYrQtr = WshDashboard.cmbQtr.Value

'Trap for user somehow not having picked a valid path...
If Not Utilities.DoesFolderOrFileExist(sSIFoldersParentPath) Then
    g_cMsgBoxHandler.SetInvalidSIFolderMsg
    g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbExclamation
    GoTo ExitPoint
End If

'Confirm/reset range of SI numbers...
Utilities.ResizeXLNameViaWshReferences _
    XLName:="coldoSINos_Dshbd", _
    RanOkRESULT:=bCalledProcedureRanOk, _
    WKSHColNmbrForVertResizing:=WshDashboard.Range("colSINos_Dshbd").Column
If Not bCalledProcedureRanOk Then Err.Raise g_lHANDLED_ERROR
Set rngSINosDshbdWs = WshDashboard.Range("coldoSINos_Dshbd")

'Trap for an empty list...
If rngSINosDshbdWs.Rows.Count = 1 And IsEmpty(rngSINosDshbdWs.Cells(1, 1)) Then
    g_cMsgBoxHandler.SetNoSIsOnDashboardMsg
    g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbExclamation
    'Stop executing this procedure...
    GoTo ExitPoint
End If
Application.StatusBar = "Retrieving Discrete Data. Please wait..."

'Remove filters on any of the sheets being accessed in this procedure...
RemoveFiltersFromSelectWshs _
    ForInitiatingQtr:=False, _
    RanOkRESULT:=bCalledProcedureRanOk
If Not bCalledProcedureRanOk Then Err.Raise g_lHANDLED_ERROR

'Before proceeding see if the user has added or deleted any columns in our _
data tables.  This procedure checks both the Federal and State worksheets. _
If the procedure finds any changes it throws a project-terminating error...
CheckStructuralIntegityOfDataTables _
    RetrievingDataForQtr:=True, _
    RanOkRESULT:=bCalledProcedureRanOk
If Not bCalledProcedureRanOk Then Err.Raise g_lHANDLED_ERROR
```

```
ReDim aDataType(0 To yNMBR_DATA_TYPES - 1)
aDataType(0) = projFederal
aDataType(1) = projState
For i = 0 To UBound(aDataType)
    MoveDataOffWkgArWshs _
        InitiatingQtr:=False, _
        WhichData:=aDataType(i), _
        WkgAreaEmptyRESULT:=bNoWkgArData, _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lHANDLED_ERROR

    If bNoWkgArData And aDataType(i) = projFederal Then _
        bFirstDataPullForQtrFed = True
    If bNoWkgArData And aDataType(i) = projState Then _
        bFirstDataPullForQtrSt = True

    SetWkgAreaWkshColHdrs _
        WhichData:=aDataType(i), _
        RanOkRESULT:=bCalledProcedureRanOk
    If Not bCalledProcedureRanOk Then Err.Raise g_lHANDLED_ERROR
Next i

'*** NOTE:  chr(92) = backslash, i.e. "\"
'Loop through the SI's and retrieve data from each external workbook...
For Each cl In rngSINosDshbdWs
    sSINmbr = cl
    'Since we don't need or want to run any Workbook_Open() code that _
    might be inside the SI Wkbs...
    Application.EnableEvents = False

    sSIFolder = sSIFoldersParentPath & sSINmbr & Chr(92)
    sSIWbNm = sYrQtr & g_sSI_WB_NM_SUFFIX
    sSIWbFullNm = sSIFolder & sSIWbNm

    'Trap for not being able to find a file...
    bSIWbkExists = Utilities.DoesFolderOrFileExist(sSIWbFullNm)
    If Not bSIWbkExists Then
        'Set boolean flag, which is checked in the Workbook_close() event...
        g_bSIWbProbs = True

        UpdateProblemFileLog _
            ProblemType:=projCannotFind, _
            ProbFilePath:=sSIFolder, _
            ProbFileName:=sSIWbNm
        'Tell user about problem and see if he wants to abort or proceed...
        g_cMsgBoxHandler.SetCannotFindFileMsg _
            FileNm:=sSIWbNm, _
            Path:=sSIFolder, _
            LogFileFullNm:=ThisWorkbook.Path & Chr(92) & _
                g_sSI_PROBLEM_WBKS_LOG
        ansContOpeningFiles = _
            g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbCritical + vbYesNo)
        If ansContOpeningFiles = vbNo Then
            GoTo ExitPoint
        Else
            GoTo LoopToNextWb
        End If
    End If

    'File exists.  Try to open...
    'Forge ahead even if/when we encounter errors opening an SI wbk...
    On Error Resume Next
    Workbooks.Open _
        Filename:=sSIWbFullNm, _
        UpdateLinks:=2, _
        ReadOnly:=True
    Err.Clear '...in case we did encounter any errors

    'Test for THE error - were we in fact able to open the SI Wbk...
    bSIWbkOpened = Utilities.IsWorkbookOpen(sSIWbNm)
    'Yes, we WANT to remain in On Error Resume Next within this customized _
    "error-handler"...
```

```
    If Not bSIWbkOpened Then '...we couldn't open the SI file
    'Set flag, which is checked in the Workbook_close() event...
        g_bSIWbProbs = True
        UpdateProblemFileLog _
            ProblemType:=projCannotOpen, _
            ProbFilePath:=sSIFolder, _
            ProbFileName:=sSIWbNm
        On Error GoTo ErrorHandler '...NOW reset
        'Tell user about problem and see if s/he wants to abort or proceed...
        g_cMsgBoxHandler.SetCannotOpenFileMsg _
            FileFullNm:=sSIWbFullNm, _
            LogFileFullNm:= _
                ThisWorkbook.Path & Chr(92) & g_sSI_PROBLEM_WBKS_LOG
        ansContOpeningFiles = _
            g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbCritical + vbYesNo)
        If ansContOpeningFiles = vbNo Then
            GoTo ExitPoint
        Else
            GoTo LoopToNextWb
        End If
    End If
    On Error GoTo ErrorHandler '...reset

    'If here we were able to find and open the file (yippee!!!)...
    ThisWorkbook.Activate
    Set wbSI = Workbooks(sSIWbNm)
    bSIWbkStructureCorrupt = False '...reset for each wb
    For i = 0 To UBound(aDataType)
        Select Case aDataType(i)
            Case projFederal
                sXLNmTblNmWhDiscrWkgAr = "tblwhdynDiscr_WkgArFed"
            Case projState
                sXLNmTblNmWhDiscrWkgAr = "tblwhdynDiscr_WkgArSt"
        End Select
        RetrieveDataFmSIWbk _
            SIWkbk:=wbSI, _
            SI:=sSINmbr, _
            WhichData:=aDataType(i), _
            WbTblsCorruptRESULT:=bSIWbkStructureCorrupt, _
            StopProcessingFilesRESULT:=bStopProcessingFiles, _
            RanOkRESULT:=bCalledProcedureRanOk
        'It seems we need to do this before diving back into State stuff...
        ThisWorkbook.Activate
        WshDashboard.Activate
        If Not bCalledProcedureRanOk Then
            UpdateProblemFileLog _
                ProblemType:=projGenlSIWbProbs, _
                ProbFilePath:=sSIFolder, _
                ProbFileName:=wbSI.Name
            g_cMsgBoxHandler.SetProblemsWorkingWithSIWbMsg _
                FileFullNm:=wbSI.FullName, _
                LogFileFullNm:= _
                    ThisWorkbook.Path & Chr(92) & g_sSI_PROBLEM_WBKS_LOG
            ansContOpeningFiles = _
                g_cMsgBoxHandler.ShowMsgBoxReturnAnswer(vbYesNo + vbInformation)
            If ansContOpeningFiles = vbNo Then bStopProcessingFiles = True
            Exit For
        End If
    Next i
    wbSI.Close savechanges:=False
    'We need to trap for instances where user has problems opening each and _
    every workbook.  (We ran into this prob the first time we stepped through _
    the "finished" code!)  If NOTHING got opened and copied into this wb then _
    we will shortly want to bail out of the remainder of the code...
    If bCalledProcedureRanOk Then bUpdatedSomeData = True
    If bStopProcessingFiles Then GoTo ExitPoint
LoopToNextWb:
Next cl

Application.EnableEvents = True
If Not bUpdatedSomeData Then GoTo ExitPoint

'Now populate the classification columns via table lookups...
```

```
CmdObjHandlers - 22

For i = 0 To UBound(aDataType)
    Select Case aDataType(i)
        Case projFederal
            Set wsWkgAr = WshWkgAreaFed
            sXLNmTblNmWhDiscrWkgAr = "tblwhdynDiscr_WkgArFed"
            bUseDblLkupFormula = bFirstDataPullForQtrFed
        Case projState
            Set wsWkgAr = WshWkgAreaSt
            sXLNmTblNmWhDiscrWkgAr = "tblwhdynDiscr_WkgArSt"
            bUseDblLkupFormula = bFirstDataPullForQtrSt
    End Select

    With wsWkgAr
        .Activate
        lVertSizeWkgArWshInclHdr = .Range(sXLNmTblNmWhDiscrWkgAr).Rows.Count
    End With

    If lVertSizeWkgArWshInclHdr > 1 Then
        PopulateWkgArWsClassfColsViaVLkups _
            WhichData:=aDataType(i), _
            NewRows:=lVertSizeWkgArWshInclHdr - 1, _
            Embed2ndLkUpUsingPrQtrKey:=bUseDblLkupFormula, _
            RanOkRESULT:=bCalledProcedureRanOk
        If Not bCalledProcedureRanOk Then Err.Raise g_lHANDLED_ERROR
    End If
Next i
WshDashboard.Activate

ExitPoint:
    On Error Resume Next
    'Must-run procedure clean-up code...
    WshDashboard.Activate
    WshDashboard.Range("A1").Select

    'Update activity log...
    If bUpdatedSomeData Then
        WshDashboard.Range("hdrLog_Dshbd").Offset(1, 0) = _
            Format(Now(), "MMM DD, YYYY H:MM AMPM")
        'Let user know we're done...
        g_cMsgBoxHandler.SetDiscrDataPopulatedMsg YrQtr:=WshDashboard.cmbQtr.Value
        g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbInformation
    End If

    Application.EnableEvents = True
    Application.StatusBar = vbNullString
    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
        'So we can step through the code which caused the error...
        Resume
    Else
        Resume ExitPoint
    End If
End Sub
'===============================================================================
'===============================================================================
'===============================================================================
'===============================================================================
'===============================================================================
'===============================================================================
'===============================================================================
```

```
'.....................................................................................
'.....................................................................................
'.......        "Written" by:          William H. White (see Comments...)
'.......        With:                  TEKSystems
'.......                               www.teksystems.com
'.......        For:                   AIG, Inc.
'.......                               Tax Technology Group
'.......                               1 WFC, 14th floor
'.......        Current contact:       william.white@aig.com
'.......                               (212) 581-5846
'.......        Permanent contact:     www.rcpconsulting.biz
'.......                               billwhite@rcpconsulting.biz
'.......                               New Providence, NJ
'.......        Module created:        January 18, 2012
'.......        Proj finished:         March 21, 2012
'.....................................................................................
'.....................................................................................


'.....................................................................................
'.........................  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_sFILE_ERROR_LOG As String = "Error.log"



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

```
ErrorHandling - 2

'=============================================================
'=================  (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:
    '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., " "
'NOTE 3:  The boolean returned by this function sets whether or you you _
wind up stepping through the results of the error.

Static sErrMsg As String
Static sFileNmOrigErr As String
Static sProcNmOrgErr As String
Dim iFile As Integer
Dim lErrNmbr As Long
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 Is = g_lUSER_CANCEL
                sErrMsg = m_sUSER_CANCELED
        Case Is = g_lMONKEY_WRENCH
                'We've thrown a deliberate error...
                sErrMsg = m_sTESTING_ERROR
        Case Is = g_lHANDLED_ERROR
                'Error in a call sub or invoked function...
                sErrMsg = m_sSUB_FUNC_ERROR
        Case Is = g_lMISMATCHED_WSHS
                'Ranges must reside on same worksheet...
                sErrMsg = m_sMISMATCHED_PARENT_WS_ERROR
        Case Is = g_lNO_DATA_IN_RNG
                sErrMsg = m_sNO_DATA_IN_DEFINED_RNG_ERROR
        Case Is = g_lSTRUCTURE_ALTERED
                'Some wksheet, table, range, etc. has wrong number of cols...
                sErrMsg = m_sSTRUCTURE_CHANGED
        Case Else
                'Do nothing - this is the originating error
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 Len(sErrMsg) = 0 Then sErrMsg = Err.Description
If Len(sProcNmOrgErr) = 0 Then sProcNmOrgErr = ProcedureName
If Len(sFileNmOrigErr) = 0 Then
    If Len(FileNm) = 0 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 Len(FileNm) = 0 Then FileNm = ThisWorkbook.Name
```

```
ErrorHandling - 3

'Get the application directory...
sPath = ThisWorkbook.Path
If StrComp(Right$(sPath, 1), Chr(92)) <> 0 Then sPath = sPath & Chr(92)

'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) & ":  " & sErrMsg

'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(sErrMsg, 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:=sErrMsg
        g_cMsgBoxHandler.ShowMsgBoxNoAnswer vbCritical
        'So that we are ready to handle the next error, clear the static _
        error message variable once we've reached the entry point...
        sErrMsg = 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 vbCritical
        End If
        If IsEntryPoint Then sErrMsg = vbNullString
        StepThroughErrorModeRESULT = False
End If
End Sub
'===============================================================================
'===============================================================================
```

```
'.....................................................................................
'.....................................................................................
'.......        Written by:             William H. White (consultant)
'.......        With:                   TEKSystems
'.......                                www.teksystems.com
'.......        For:                    AIG, Inc.
'.......                                Tax Technology Group
'.......                                1 WFC, 14th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 581-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Provicence, NJ
'.......        Module created:         January 17, 2012
'.......        Proj finished:          March 21, 2012
'.....................................................................................
'.....................................................................................

'.....................................................................................
'..............  REFERENCE LIBRARIES EMPLOYED/REQUIRED BY PROJECT  .............
'Microsoft Shell Control and Automation
'Microsoft Scripting Runtime
'.....................................................................................
'.....................................................................................

'.....................................................................................
'............................  Class-level COMMENTS  ............................
'This module contains all "normal" global variable declarations
'*** Because of the number of the constants I have defined, ALL COLUMN NUMBER _
AND COLUMN NAME CONSTANSTS FOR THE COLUMN HEADERS IN THE SI WKBK FILES ARE _
CONTAINED IN THE SIWbkColGlobals MODULE!!!
'.....................................................................................
'.....................................................................................




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


'=====================================================================================
'=====================================================================================
'**************************************
'CONSTANTS
'**************************************
'Bytes [Note:  range = 0 to 255]...
Public Const g_yWA_CLASSFCTN_COLS As Byte = 6
Public Const g_yWA_HDR_DATA_COLS As Byte = 3
Public Const g_ySIWB_DISCRWS_COLS As Byte = 68
Public Const g_ySIWB_DISCRWS_HDR_ROW As Byte = 15

Public Const g_yWACLASSFCOL_ETR As Byte = 1
Public Const g_yWACLASSFCOL_ETR_OTHR As Byte = 2
Public Const g_yWACLASSFCOL_HI_LVL As Byte = 3
Public Const g_yWACLASSFCOL_SUMM As Byte = 4
Public Const g_yWACLASSFCOL_SEG As Byte = 5
Public Const g_yWACLASSFCOL_DISC_OP As Byte = 6

Public Const g_yWAHDRIDCOL_PER As Byte = 7
Public Const g_yWAHDRIDCOL_SI As Byte = 8
Public Const g_yWAHDRIDCOL_SI_DESCR As Byte = 9


'Longs...
'[NOTE:  numbers 1024 through 65535 are available for custom errors]
Public Const g_lHANDLED_ERROR As Long = 9999
Public Const g_lMONKEY_WRENCH As Long = 9876
Public Const g_lMISMATCHED_WSHS As Long = 9998
```

```
Public Const g_lNO_DATA_IN_RNG As Long = 9997
Public Const g_lSTRUCTURE_ALTERED As Long = 9996
Public Const g_lUSER_CANCEL As Long = 18

'Strings...
Public Const g_sAPP_TITLE As String = "DISCRETE ITEMS CONSOLIDATION"
Public Const g_sWSH_PASSWORD As String = "mtw2012" '"fas!09"
Public Const g_sSI_WB_NM_SUFFIX As String = "_Discrete_Item_Template.xls"
Public Const g_sSI_PROBLEM_WBKS_LOG As String = "DiscrData_ProbWbks.log"

Public Const g_sWACLASSFCOL_ETR As String = "ETR"
Public Const g_sWACLASSFCOL_ETR_OTHR As String = "ETR_Other"
Public Const g_sWACLASSFCOL_HI_LVL As String = "High_Level"
Public Const g_sWACLASSFCOL_SUMM As String = "Summary"
Public Const g_sWACLASSFCOL_SEG As String = "Segment"
Public Const g_sWACLASSFCOL_DISC_OP As String = "DISC_OP"

Public Const g_sWAHDRIDCOL_PER As String = "Period"
Public Const g_sWAHDRIDCOL_SI As String = "SI No"
Public Const g_sWAHDRIDCOL_SI_DESCR As String = "SI Description"


'*************************************
'VARIABLES
'*************************************
'Classes...
Public g_cMsgBoxHandler As MsgBoxHandler

'Strings...
Public g_sYrQtr As String
Public g_sXLFilePath As String

'Booleans (default = FALSE)...
Public g_bStepThruMode As Boolean
Public g_bStepThruWsChangeEventMode As Boolean
Public g_bDebugMode As Boolean
Public g_bSIWbProbs As Boolean


'*************************************
'TYPES
'*************************************
Public Type TypColNmNoPairs
    typColNo As Byte
    typColNm As String
End Type


'*************************************
'ENUMERATIONS
'*************************************
Public Enum ProjDataType
    projFederal = 1
    projState = 2
End Enum

Public Enum ProjFileProb
    'projNoProbs = 0
    projCannotFind = 1
    projCannotOpen = 2
    projSIWbDiscrWsCols = 4
    projGenlSIWbProbs = 8
    'projBothProbs = 3
End Enum
```

SIWbkColGlobals - 1

```
Option Explicit

'NOTE:  These declarations were generated by doing a copy/transpose _
of the column headers into an Excel spreadsheet and then using a little _
VBA elbow grease build the statements...


Public Const g_sSIWSCOL_KEY As String = "Key"
Public Const g_ySIWSCOL_KEY As Byte = 1

Public Const g_sSIWSCOL_PR_QTR_KEY As String = "Prior_Qtr_Key"
Public Const g_ySIWSCOL_PR_QTR_KEY As Byte = 2

Public Const g_sSIWSCOL_DISCR_ITM_DESC As String = "Discrete Item Description"
Public Const g_ySIWSCOL_DISCR_ITM_DESC As Byte = 3

Public Const g_sSIWSCOL_DISCR_ITM_CAT As String = "Discrete Item Category"
Public Const g_ySIWSCOL_DISCR_ITM_CAT As Byte = 4

Public Const g_sSIWSCOL_TAX_TYPE As String = "Type of Tax"
Public Const g_ySIWSCOL_TAX_TYPE As Byte = 5

Public Const g_sSIWSCOL_ORD_RCG As String = "Ordinary / RCG"
Public Const g_ySIWSCOL_ORD_RCG As Byte = 6

Public Const g_sSIWSCOL_DISCR_TYPE As String = "Type of Discrete"
Public Const g_ySIWSCOL_DISCR_TYPE As Byte = 7

Public Const g_sSIWSCOL_COMP_NM As String = "Company Name"
Public Const g_ySIWSCOL_COMP_NM As Byte = 8

Public Const g_sSIWSCOL_RO_NMBR As String = "RO Number"
Public Const g_ySIWSCOL_RO_NMBR As Byte = 9

Public Const g_sSIWSCOL_CNTRY As String = "Country of Operation"
Public Const g_ySIWSCOL_CNTRY As Byte = 10

Public Const g_sSIWSCOL_APB_STATUS As String = "APB Status"
Public Const g_ySIWSCOL_APB_STATUS As Byte = 11

Public Const g_sSIWSCOL_RPTG_UNT_US_MEMB_NON As String = "Reporting_Unit US Member / Non-Member"
Public Const g_ySIWSCOL_RPTG_UNT_US_MEMB_NON As Byte = 12

Public Const g_sSIWSCOL_IN_OUT_PER_Q1 As String = "Q1 In Period / Out of Period"
Public Const g_ySIWSCOL_IN_OUT_PER_Q1 As Byte = 13

Public Const g_sSIWSCOL_IN_OUT_SCOPE_Q1 As String = "Q1 In Scope/Out of Scope"
Public Const g_ySIWSCOL_IN_OUT_SCOPE_Q1 As Byte = 14

Public Const g_sSIWSCOL_FX_RT_Q1 As String = "Q1 FX Rate"
Public Const g_ySIWSCOL_FX_RT_Q1 As Byte = 15

Public Const g_sSIWSCOL_US_MEMB_LC_Q1 As String = "Q1 U.S. Member Amount in LC"
Public Const g_ySIWSCOL_US_MEMB_LC_Q1 As Byte = 16

Public Const g_sSIWSCOL_US_MEMB_USD_Q1 As String = "Q1 U.S. Member Amount in USD"
Public Const g_ySIWSCOL_US_MEMB_USD_Q1 As Byte = 17

Public Const g_sSIWSCOL_US_MEMB_FTC_USD_Q1 As String = "Q1 U.S. Member FTC in USD"
Public Const g_ySIWSCOL_US_MEMB_FTC_USD_Q1 As Byte = 18

Public Const g_sSIWSCOL_US_MEMB_NET_FTC_USD_Q1 As String = "Q1 U.S. Member Amount Net of FTC in USD"
Public Const g_ySIWSCOL_US_MEMB_NET_FTC_USD_Q1 As Byte = 19

Public Const g_sSIWSCOL_NONMEMB_LC_Q1 As String = "Q1 Non-Member Amount in LC"
Public Const g_ySIWSCOL_NONMEMB_LC_Q1 As Byte = 20

Public Const g_sSIWSCOL_NONMEMB_USD_Q1 As String = "Q1 Non-Member Amount in USD"
Public Const g_ySIWSCOL_NONMEMB_USD_Q1 As Byte = 21

Public Const g_sSIWSCOL_CORP_TX_APPR_REQ_Q1 As String = "Q1 Corporate Tax Approval required"
Public Const g_ySIWSCOL_CORP_TX_APPR_REQ_Q1 As Byte = 22
```

```
Public Const g_sSIWSCOL_ATOI_IMPCT_USD_Q1 As String = "Q1 ATOI Impact in USD"
Public Const g_ySIWSCOL_ATOI_IMPCT_USD_Q1 As Byte = 23

Public Const g_sSIWSCOL_US_GAAP_IMPCT_Q1 As String = "Q1 U.S. GAAP Impact"
Public Const g_ySIWSCOL_US_GAAP_IMPCT_Q1 As Byte = 24

Public Const g_sSIWSCOL_IN_OUT_PER_Q2 As String = "Q2 In Period / Out of Period"
Public Const g_ySIWSCOL_IN_OUT_PER_Q2 As Byte = 25

Public Const g_sSIWSCOL_IN_OUT_SCOPE_Q2 As String = "Q2 In Scope/Out of Scope"
Public Const g_ySIWSCOL_IN_OUT_SCOPE_Q2 As Byte = 26

Public Const g_sSIWSCOL_FX_RT_Q2 As String = "Q2 FX Rate"
Public Const g_ySIWSCOL_FX_RT_Q2 As Byte = 27

Public Const g_sSIWSCOL_US_MEMB_LC_Q2 As String = "Q2 U.S. Member Amount in LC"
Public Const g_ySIWSCOL_US_MEMB_LC_Q2 As Byte = 28

Public Const g_sSIWSCOL_US_MEMB_USD_Q2 As String = "Q2 U.S. Member Amount in USD"
Public Const g_ySIWSCOL_US_MEMB_USD_Q2 As Byte = 29

Public Const g_sSIWSCOL_US_MEMB_FTC_USD_Q2 As String = "Q2 U.S. Member FTC in USD"
Public Const g_ySIWSCOL_US_MEMB_FTC_USD_Q2 As Byte = 30

Public Const g_sSIWSCOL_US_MEMB_NET_FTC_USD_Q2 As String = "Q2 U.S. Member Amount Net of FTC in USD"
Public Const g_ySIWSCOL_US_MEMB_NET_FTC_USD_Q2 As Byte = 31

Public Const g_sSIWSCOL_NONMEMB_LC_Q2 As String = "Q2 Non-Member Amount in LC"
Public Const g_ySIWSCOL_NONMEMB_LC_Q2 As Byte = 32

Public Const g_sSIWSCOL_NONMEMB_USD_Q2 As String = "Q2 Non-Member Amount in USD"
Public Const g_ySIWSCOL_NONMEMB_USD_Q2 As Byte = 33

Public Const g_sSIWSCOL_CORP_TX_APPR_REQ_Q2 As String = "Q2 Corporate Tax Approval required"
Public Const g_ySIWSCOL_CORP_TX_APPR_REQ_Q2 As Byte = 34

Public Const g_sSIWSCOL_ATOI_IMPCT_USD_Q2 As String = "Q2 ATOI Impact in USD"
Public Const g_ySIWSCOL_ATOI_IMPCT_USD_Q2 As Byte = 35

Public Const g_sSIWSCOL_US_GAAP_IMPCT_Q2 As String = "Q2 U.S. GAAP Impact"
Public Const g_ySIWSCOL_US_GAAP_IMPCT_Q2 As Byte = 36

Public Const g_sSIWSCOL_IN_OUT_PER_Q3 As String = "Q3 In Period / Out of Period"
Public Const g_ySIWSCOL_IN_OUT_PER_Q3 As Byte = 37

Public Const g_sSIWSCOL_IN_OUT_SCOPE_Q3 As String = "Q3 In Scope/Out of Scope"
Public Const g_ySIWSCOL_IN_OUT_SCOPE_Q3 As Byte = 38

Public Const g_sSIWSCOL_FX_RT_Q3 As String = "Q3 FX Rate"
Public Const g_ySIWSCOL_FX_RT_Q3 As Byte = 39

Public Const g_sSIWSCOL_US_MEMB_LC_Q3 As String = "Q3 U.S. Member Amount in LC"
Public Const g_ySIWSCOL_US_MEMB_LC_Q3 As Byte = 40

Public Const g_sSIWSCOL_US_MEMB_USD_Q3 As String = "Q3 U.S. Member Amount in USD"
Public Const g_ySIWSCOL_US_MEMB_USD_Q3 As Byte = 41

Public Const g_sSIWSCOL_US_MEMB_FTC_USD_Q3 As String = "Q3 U.S. Member FTC in USD"
Public Const g_ySIWSCOL_US_MEMB_FTC_USD_Q3 As Byte = 42

Public Const g_sSIWSCOL_US_MEMB_NET_FTC_USD_Q3 As String = "Q3 U.S. Member Amount Net of FTC in USD"
Public Const g_ySIWSCOL_US_MEMB_NET_FTC_USD_Q3 As Byte = 43

Public Const g_sSIWSCOL_NONMEMB_LC_Q3 As String = "Q3 Non-Member Amount in LC"
Public Const g_ySIWSCOL_NONMEMB_LC_Q3 As Byte = 44

Public Const g_sSIWSCOL_NONMEMB_USD_Q3 As String = "Q3 Non-Member Amount in USD"
Public Const g_ySIWSCOL_NONMEMB_USD_Q3 As Byte = 45

Public Const g_sSIWSCOL_CORP_TX_APPR_REQ_Q3 As String = "Q3 Corporate Tax Approval required"
Public Const g_ySIWSCOL_CORP_TX_APPR_REQ_Q3 As Byte = 46

Public Const g_sSIWSCOL_ATOI_IMPCT_USD_Q3 As String = "Q3 ATOI Impact in USD"
```

```
Public Const g_ySIWSCOL_ATOI_IMPCT_USD_Q3 As Byte = 47

Public Const g_sSIWSCOL_US_GAAP_IMPCT_Q3 As String = "Q3 U.S. GAAP Impact"
Public Const g_ySIWSCOL_US_GAAP_IMPCT_Q3 As Byte = 48

Public Const g_sSIWSCOL_IN_OUT_PER_Q4 As String = "Q4 In Period / Out of Period"
Public Const g_ySIWSCOL_IN_OUT_PER_Q4 As Byte = 49

Public Const g_sSIWSCOL_IN_OUT_SCOPE_Q4 As String = "Q4 In Scope/Out of Scope"
Public Const g_ySIWSCOL_IN_OUT_SCOPE_Q4 As Byte = 50

Public Const g_sSIWSCOL_FX_RT_Q4 As String = "Q4 FX Rate"
Public Const g_ySIWSCOL_FX_RT_Q4 As Byte = 51

Public Const g_sSIWSCOL_US_MEMB_LC_Q4 As String = "Q4 U.S. Member Amount in LC"
Public Const g_ySIWSCOL_US_MEMB_LC_Q4 As Byte = 52

Public Const g_sSIWSCOL_US_MEMB_USD_Q4 As String = "Q4 U.S. Member Amount in USD"
Public Const g_ySIWSCOL_US_MEMB_USD_Q4 As Byte = 53

Public Const g_sSIWSCOL_US_MEMB_FTC_USD_Q4 As String = "Q4 U.S. Member FTC in USD"
Public Const g_ySIWSCOL_US_MEMB_FTC_USD_Q4 As Byte = 54

Public Const g_sSIWSCOL_US_MEMB_NET_FTC_USD_Q4 As String = "Q4 U.S. Member Amount Net of FTC in USD"
Public Const g_ySIWSCOL_US_MEMB_NET_FTC_USD_Q4 As Byte = 55

Public Const g_sSIWSCOL_NONMEMB_LC_Q4 As String = "Q4 Non-Member Amount in LC"
Public Const g_ySIWSCOL_NONMEMB_LC_Q4 As Byte = 56

Public Const g_sSIWSCOL_NONMEMB_USD_Q4 As String = "Q4 Non-Member Amount in USD"
Public Const g_ySIWSCOL_NONMEMB_USD_Q4 As Byte = 57

Public Const g_sSIWSCOL_CORP_TX_APPR_REQ_Q4 As String = "Q4 Corporate Tax Approval required"
Public Const g_ySIWSCOL_CORP_TX_APPR_REQ_Q4 As Byte = 58

Public Const g_sSIWSCOL_ATOI_IMPCT_USD_Q4 As String = "Q4 ATOI Impact in USD"
Public Const g_ySIWSCOL_ATOI_IMPCT_USD_Q4 As Byte = 59

Public Const g_sSIWSCOL_US_GAAP_IMPCT_Q4 As String = "Q4 U.S. GAAP Impact"
Public Const g_ySIWSCOL_US_GAAP_IMPCT_Q4 As Byte = 60

Public Const g_sSIWSCOL_US_MEMB_LC_YTD As String = "YTD U.S. Member Amount in LC"
Public Const g_ySIWSCOL_US_MEMB_LC_YTD As Byte = 61

Public Const g_sSIWSCOL_US_MEMB_USD_YTD As String = "YTD U.S. Member Amount in USD"
Public Const g_ySIWSCOL_US_MEMB_USD_YTD As Byte = 62

Public Const g_sSIWSCOL_US_MEMB_FTC_USD_YTD As String = "YTD U.S. Member FTC in USD"
Public Const g_ySIWSCOL_US_MEMB_FTC_USD_YTD As Byte = 63

Public Const g_sSIWSCOL_US_MEMB_NET_FTC_USD_YTD As String = "YTD U.S. Member Amount Net of FTC in USD"
Public Const g_ySIWSCOL_US_MEMB_NET_FTC_USD_YTD As Byte = 64

Public Const g_sSIWSCOL_NONMEMB_LC_YTD As String = "YTD Non-Member Amount in LC"
Public Const g_ySIWSCOL_NONMEMB_LC_YTD As Byte = 65

Public Const g_sSIWSCOL_NONMEMB_USD_YTD As String = "YTD Non-Member Amount in USD"
Public Const g_ySIWSCOL_NONMEMB_USD_YTD As Byte = 66

Public Const g_sSIWSCOL_ATOI_IMPCT_USD_YTD As String = "YTD ATOI Impact in USD"
Public Const g_ySIWSCOL_ATOI_IMPCT_USD_YTD As Byte = 67

Public Const g_sSIWSCOL_US_GAAP_IMPCT_YTD As String = "YTD U.S. GAAP Impact"
Public Const g_ySIWSCOL_US_GAAP_IMPCT_YTD As Byte = 68
```

```
'.....................................................................................
'.....................................................................................
'.......        Written by:             William H. White (consultant)
'.......        With:                   TEKSystems
'.......                                www.teksystems.com
'.......        For:                    AIG, Inc.
'.......                                Tax Technology Group
'.......                                1 WFC, 14th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 581-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         January 18, 2012
'.......        Proj finished:          March 21, 2012
'.....................................................................................
'.....................................................................................


'.....................................................................................
'.........................  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
```

```
Utilities - 2

'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 Function FindLastRowInColumn( _
    ByRef wsParent As Worksheet, _
    ByVal WshColNmbr As Long, _
    ByRef RanOkRESULT As Boolean, _
    Optional ParentWkbk As Workbook) As Long
'Error-handling declarations...
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "FindLastRowInColumn()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wbOrig As Workbook
Dim wsOrig As Worksheet
Dim rngOrig As Range
Dim rngCol As Range
Dim rngLastWsCell As Range
Dim rngLastCellInCol As Range
Dim lLastRow As Long

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

'Capture current status...
Set wbOrig = ActiveWorkbook
Set wsOrig = ActiveSheet
Set rngOrig = Selection

'Smurf over to the relevant workbook, if it's been supplied...
If Not ParentWkbk Is Nothing Then ParentWkbk.Activate

Set rngCol = wsParent.Columns(WshColNmbr)

'Start to find last cell...
Set rngLastWsCell = wsParent.Cells.SpecialCells(xlCellTypeLastCell)
lLastRow = rngLastWsCell.Row

Set rngLastCellInCol = wsParent.Cells(lLastRow, WshColNmbr)
If IsEmpty(rngLastCellInCol) Then
    wsParent.Activate
    rngLastCellInCol.End(xlUp).Select
    Set rngLastCellInCol = Selection
End If

'Viola...
FindLastRowInColumn = rngLastCellInCol.Row

ExitPoint:
On Error Resume Next
'Must-run procedure/function clean-up code...
wbOrig.Activate
wsOrig.Activate
rngOrig.Select
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 FindLastColInRow( _
    ByRef wsParent As Worksheet, _
    ByVal WshRowNmbr As Long, _
    ByRef RanOkRESULT As Boolean, _
    Optional ParentWkbk As Workbook) As Long
'Error-handling declarations...
Const bFUNCTION_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "FindLastColInRow()"
Dim sErrSourceFileFullNm As String
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wbOrig As Workbook
Dim wsOrig As Worksheet
Dim rngOrig As Range
Dim rngRow As Range
Dim rngLastWsCell As Range
Dim rngLastCellInRow As Range
Dim lLastCol As Long

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

'Capture current status...
Set wbOrig = ActiveWorkbook
Set wsOrig = ActiveSheet
Set rngOrig = Selection

'Smurf over to the relevant workbook, if it's been supplied...
If Not ParentWkbk Is Nothing Then
    ParentWkbk.Activate
    sErrSourceFileFullNm = ParentWkbk.FullName
End If

Set rngRow = wsParent.Columns(WshRowNmbr)

'Start to find last cell...
Set rngLastWsCell = wsParent.Cells.SpecialCells(xlCellTypeLastCell)
lLastCol = rngLastWsCell.Column

Set rngLastCellInRow = wsParent.Cells(WshRowNmbr, lLastCol)
If IsEmpty(rngLastCellInRow) Then
    wsParent.Activate
    rngLastCellInRow.End(xlToLeft).Select
    Set rngLastCellInRow = Selection
End If

'Viola...
FindLastColInRow = rngLastCellInRow.Column

ExitPoint:
On Error Resume Next
'Must-run procedure/function clean-up code...
wbOrig.Activate
wsOrig.Activate
```

```
rngOrig.Select
Exit Function

ErrorHandler:
    RanOkRESULT = False
    Dim bRetraceErrorMode As Boolean
    ErrorHandling.CentralErrorHandler _
        ModuleName:=m_sMODULE_NAME, _
        ProcedureName:=sSOURCE, _
        StepThroughErrorModeRESULT:=bRetraceErrorMode, _
        FileNm:=sErrSourceFileFullNm, _
        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 VerifyHorizDimensXLNamedRegion( _
    ByVal XLName As String, _
    ByVal RANGERowToSizeOn As Long, _
    ByVal ExpectedNoCols As Long, _
    ByRef RanOkRESULT As Boolean, _
    Optional ParentWkbk As Workbook, _
    Optional WshNameIfXLNmWshScoped As String) As Boolean

'*******************
'NOTE:  This function DOES NOT RAISE AN ERROR if it is found that the _
horizontal dimensions of an XL-named region are out of whack!!! Rather, this _
function simply returns a value of FALSE.
'Chosing how to respond to having out-of-whack horizontal dimensions of an _
XL-named region IS THE RESPONSIBILITY OF THE PROCEDURE WHICH INVOKES THIS _
FUNCTION!!!
'IF having out-of-whack horizontal dimensions needs to be considered a _
catastrophic error the recommended procedure is to use the following _
(pseudo-) code inside the calling procedure (assume XLName="rgnTest", _
row inside range to size on = 1, and number of expected columns = 10)...

    'Sub CallingProcedure()
    '...other error-handling variable declarations
    'Dim bSubRoutineRanOK As Boolean
    'Dim bHorizDimensOk As Boolean
    '...other operative code variable declarations

    'bHorizDimensOk=VerifyHorizDimensXLNamedRegion("rgnTest",1,10,bSubRoutineRanOK)
    'If Not bCalledProcedureRanOk Then Err.Raise g_lHANDLED_ERROR
    '...NOW raise a project-terminating error if the table structure is off...
    'If Not bHorizDimensOk Then Err.Raise g_lSTRUCTURE_ALTERED

    '...balance of operative and standard error-handling code
    'End Sub
'This code example of course assumes the use of a/our centralized _
error-handling code and mechanisms
'*******************


'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "VerifyHorizDimensXLNamedRegion()"
Dim sErrSourceFileFullNm As String
'''Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wbOrig As Workbook
Dim wsOrig As Worksheet
Dim rngOrig As Range
Dim wsParent As Worksheet
Dim rngNamed As Range
Dim lWsRowOfRangeRowToSizeOn As Long
Dim lLastColUsedRange As Long
Dim lFirstColXLNamedRng As Long
```

```
Dim lLastColXLNamedRng As Long
Dim lNoColsXLNamedRng As Long

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

'Capture existing state...
Set wbOrig = ActiveWorkbook
Set wsOrig = ActiveSheet
Set rngOrig = Selection

'Smurf over to the relevant workbook, if it's been supplied...
If Not ParentWkbk Is Nothing Then
    sErrSourceFileFullNm = ParentWkbk.FullName
    ParentWkbk.Activate
End If

'Activate the appropriate worksheet, if appropriate...
If Not WshNameIfXLNmWshScoped = vbNullString Then
    Set wsParent = ActiveWorkbook.Worksheets(WshNameIfXLNmWshScoped)
    Set rngNamed = wsParent.Range(XLName)
Else
    Set rngNamed = Names(XLName).RefersToRange
    Set wsParent = rngNamed.Parent
End If

lWsRowOfRangeRowToSizeOn = _
    wsParent.Range(XLName).Cells(RANGERowToSizeOn, 1).Row
lFirstColXLNamedRng = wsParent.Range(XLName).Column
lLastColUsedRange = wsParent.UsedRange.Columns.Count

If Not IsEmpty(wsParent.Cells(lWsRowOfRangeRowToSizeOn, lLastColUsedRange)) Then
    lNoColsXLNamedRng = 1 + lLastColUsedRange - lFirstColXLNamedRng
Else '...cell in "farthest" RH edge of named range's 1st row is empty
    Dim rngRHMostCell As Range
    Set rngRHMostCell = wsParent.Cells(lWsRowOfRangeRowToSizeOn, lLastColUsedRange)
    wsParent.Activate '...MUST do this before selecting cell
    rngRHMostCell.End(xlToLeft).Select
    lNoColsXLNamedRng = 1 + Selection.Column - lFirstColXLNamedRng
End If

If lNoColsXLNamedRng <> ExpectedNoCols Then
    VerifyHorizDimensXLNamedRegion = False
Else
    VerifyHorizDimensXLNamedRegion = True
End If


ExitPoint:
On Error Resume Next
'Must-run procedure/function clean-up code...
wbOrig.Activate
wsOrig.Activate
rngOrig.Select
Exit Function

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

```
Utilities - 6

End Function
'============================================================================
Public Function VerifyHorizDimensWshRegion( _
    ByVal WshName As String, _
    ByVal FirstWshColOfRegion, _
    ByVal WshRowToSizeOn As Long, _
    ByVal ExpectedNoCols As Long, _
    ByRef RanOkRESULT As Boolean, _
    Optional ParentWb As Workbook) As Boolean

'*******************
'See NOTE under VerifyHorizDimensXLNamedRegion()!!
'*******************

'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "VerifyHorizDimensWshRegion()"
'''Dim bCalledProcedureRanOk As Boolean
Dim sErrSourceFileFullNm As String
'Operating code declarations...
Dim wbOrig As Workbook
Dim wsOrig As Worksheet
Dim rngOrig As Range
Dim wsToCheck As Worksheet
Dim rngRHMostCellTargetRow As Range
Dim lLastColUsedRange As Long
Dim lLastColRgn As Long
Dim lNoColsRgn As Long

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

'Capture existing state...
Set wbOrig = ActiveWorkbook
Set wsOrig = ActiveSheet
Set rngOrig = Selection

'Smurf over to the relevant workbook, if it's been supplied...
If Not ParentWb Is Nothing Then
    ParentWb.Activate
    'In case there's a problem...
    sErrSourceFileFullNm = ParentWb.FullName
End If

'Identify the appropriate worksheet...
Set wsToCheck = ParentWb.Worksheets(WshName)

'Do a preliminary check...
lLastColUsedRange = wsToCheck.UsedRange.SpecialCells(xlCellTypeLastCell).Column
If (Not IsEmpty(wsToCheck.Cells(WshRowToSizeOn, lLastColUsedRange))) And _
    (1 + lLastColUsedRange - FirstWshColOfRegion = ExpectedNoCols) Then
        'We're good!...
        VerifyHorizDimensWshRegion = True
        GoTo ExitPoint
End If

'If here we still have to find the RH-most non-empty cell in the target row _
(multiple steps)...
wsToCheck.Activate
Set rngRHMostCellTargetRow = wsToCheck.Cells(WshRowToSizeOn, lLastColUsedRange)
rngRHMostCellTargetRow.End(xlToLeft).Select
Set rngRHMostCellTargetRow = Selection

'Now see if we're good...
lLastColRgn = rngRHMostCellTargetRow.Column
If 1 + lLastColRgn - FirstWshColOfRegion = ExpectedNoCols Then
    VerifyHorizDimensWshRegion = True
Else
    VerifyHorizDimensWshRegion = False
End If
```

```
Utilities - 7

ExitPoint:
On Error Resume Next
'Must-run procedure/function clean-up code...
wbOrig.Activate
wsOrig.Activate
rngOrig.Select
Exit Function

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



'============================================================
'==================  (Public) PROCEDURES  ==================
'============================================================
'================================================================================
Public Sub ProtectWsh(ByVal WsToProtect As Worksheet, _
    ByRef RanOkRESULT As Boolean)
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "ProtectWsh()"


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

'CODE HERE...
WsToProtect.Protect _
    Password:=g_sWSH_PASSWORD, _
    contents:=True, _
    userinterfaceonly:=False

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 CreateXLNameForSingleColDataRange( _
    ByVal XLNameToDefine As String, _
```

```
    ByVal HeaderCell As Range, _
    ByVal FullColumn As Range, _
    ByVal IncludeHdrCellInDefinedRange As Boolean, _
    ByVal ThrowProjTerminatingErrIfDataRangeEmpty As Boolean, _
    ByRef RanOkRESULT As Boolean, _
    Optional ParentWkbk As Workbook)
'***NOTE:  If the data range is empty but the user does NOT want us to throw _
an error in that case, we will define a single cell as the data range.
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "CreateXLNameForSingleColDataRange()"
'''Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wbOrig As Workbook
Dim wsOrig As Worksheet
Dim wsParent As Worksheet
Dim rngOrig As Range
Dim rngToName As Range
Dim rngLastDataCell As Range
Dim lHeaderRow As Long
Dim lBottomRow As Long
Dim yRowOffsetFromHdrCell As Byte

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

'Capture current status...
Set wbOrig = ActiveWorkbook
Set wsOrig = ActiveSheet
Set rngOrig = Selection

'Smurf over to the relevant workbook, if it's been supplied...
If Not ParentWkbk Is Nothing Then ParentWkbk.Activate

'Throw an error if the two parameter ranges are not on the same worksheet...
If StrComp(HeaderCell.Parent.Name, FullColumn.Parent.Name) <> 0 Then _
    Err.Raise g_lMISMATCHED_WSHS

'Define some variables...
Set wsParent = FullColumn.Parent
lHeaderRow = HeaderCell.Row

'*****
'NOTE:  SpecialCells(xlCellTypeLastCell) apparently does NOT give you the _
last cell of a specified range.  Rather, it defaults to giving you the last _
cell of the entire worksheet.
'*****
'Find last data cell on the worksheet...
Set rngLastDataCell = wsParent.Cells.SpecialCells(xlCellTypeLastCell)
lBottomRow = rngLastDataCell.Row '...preliminary
'NOW redefine rngLastDataCell to be the last cell inside the column...
Set rngLastDataCell = FullColumn.Cells(lBottomRow, 1)

'Trap for cases where the "LastCell" is, in fact, empty...
If IsEmpty(rngLastDataCell) And lBottomRow > lHeaderRow Then
    wsParent.Activate
    rngLastDataCell.End(xlUp).Select
    Set rngLastDataCell = Selection
    lBottomRow = rngLastDataCell.Row '...NOW we have what we want
End If

'If there's no data in the range...
If lHeaderRow = lBottomRow Then
    If ThrowProjTerminatingErrIfDataRangeEmpty Then
        Err.Raise g_lNO_DATA_IN_RNG
    Else
        lBottomRow = lHeaderRow + 1 '...grab the empty cell below the header
    End If
End If

'If we ARE including a header cell then the offset remains at 0...
If Not IncludeHdrCellInDefinedRange Then yRowOffsetFromHdrCell = 1
```

```
'Now define the range (2 steps)...
Set rngToName = _
    HeaderCell.Resize(1 + lBottomRow - lHeaderRow - yRowOffsetFromHdrCell)
Set rngToName = rngToName.Offset(yRowOffsetFromHdrCell)
ThisWorkbook.Names.Add _
    Name:=XLNameToDefine, _
    RefersTo:="=" & rngToName.Parent.Name & "!" & rngToName.Address


ExitPoint:
On Error Resume Next
'Must-run procedure/function clean-up code...
wbOrig.Activate
wsOrig.Activate
rngOrig.Select
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 ResizeXLNameViaWshReferences(ByVal XLName As String, _
    ByRef RanOkRESULT As Boolean, _
    Optional ParentWkbk As Workbook, _
    Optional WshNameIfXLNmWshScoped As String, _
    Optional WKSHColNmbrForVertResizing As Long, _
    Optional WKSHRowNmbrForHorizResizing As Long)
'*********
'NOTE:  This procedure is designed to return minimum dimensions of 1 row and _
1 column, even if the named region is empty.
'*********
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "ResizeXLNameViaWshReferences()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wbOrig As Workbook
Dim wsOrig As Worksheet
Dim wsParent As Worksheet
Dim rngOrig As Range
Dim rngNamedRange As Range
Dim rngLastWsCell As Range
Dim rngLastColCell As Range
Dim rngLastRowCell As Range
Dim lWsRowNmbrLastCell As Long
Dim lWsColNmbrLastCell As Long
Dim lWsRowNmbrRangeTopCell As Long
Dim lWsColNmbrRangeTopCell As Long

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

'Capture current status...
Set wbOrig = ActiveWorkbook
Set wsOrig = ActiveSheet
Set rngOrig = Selection

'Smurf over to the relevant workbook, if it's been supplied...
If Not ParentWkbk Is Nothing Then ParentWkbk.Activate
```

```
If WshNameIfXLNmWshScoped = vbNullString Then
    Set rngNamedRange = Names(XLName).RefersToRange
    Set wsParent = rngNamedRange.Parent
Else
    Set wsParent = ActiveWorkbook.Worksheets(WshNameIfXLNmWshScoped)
    Set rngNamedRange = wsParent.Range(XLName)
End If

Set rngLastWsCell = wsParent.Cells.SpecialCells(xlCellTypeLastCell)

lWsRowNmbrRangeTopCell = rngNamedRange.Row
lWsColNmbrRangeTopCell = rngNamedRange.Column

'Handle vertical resizing...
If WKSHColNmbrForVertResizing > 0 Then
    'Preliminary definitions...
    lWsRowNmbrLastCell = rngLastWsCell.Row
    Set rngLastColCell = _
        wsParent.Cells(lWsRowNmbrLastCell, WKSHColNmbrForVertResizing)
    'If the "bottom" cell is empty we have to find the true bottom cell _
    and then vertically resize/redefine the range/Name...
    If IsEmpty(rngLastColCell) Then
        'MUST activate ws before we can activate cell...
        wsParent.Activate
        rngLastColCell.End(xlUp).Select
        Set rngLastColCell = Selection '...TRUE bottom cell
        lWsRowNmbrLastCell = rngLastColCell.Row '...TRUE bottom row
    End If
    'Make sure we have at least one row...
    If lWsRowNmbrLastCell < lWsRowNmbrRangeTopCell Then _
        lWsRowNmbrLastCell = lWsRowNmbrRangeTopCell
    Set rngNamedRange = _
        rngNamedRange.Resize(1 + lWsRowNmbrLastCell - lWsRowNmbrRangeTopCell)
End If

'Handle horizontal resizing...
If WKSHRowNmbrForHorizResizing > 0 Then
    'Preliminary definitions...
    lWsColNmbrLastCell = rngLastWsCell.Column
    Set rngLastRowCell = _
        wsParent.Cells(WKSHRowNmbrForHorizResizing, lWsColNmbrLastCell)
    'If the "right-most" cell is empty we have to find the true right-most _
    cell and then horizontally resize/redefine the range/Name...
    If IsEmpty(rngLastRowCell) Then
        If WKSHColNmbrForVertResizing = 0 Then '...didn't necessarily do this
            'MUST activate ws before we can activate cell...
            wsParent.Activate
        End If
        rngLastRowCell.End(xlToLeft).Select
        Set rngLastRowCell = Selection '...TRUE R-most cell
        lWsColNmbrLastCell = rngLastRowCell.Column '...TRUE R-most column
    End If
    'Make sure we have at least one column...
    If lWsColNmbrLastCell < lWsColNmbrRangeTopCell Then _
        lWsColNmbrLastCell = lWsColNmbrRangeTopCell
    Set rngNamedRange = _
        rngNamedRange.Resize(, 1 + lWsColNmbrLastCell - lWsColNmbrRangeTopCell)
End If

'Now redefine the XL name...
ActiveWorkbook.Names.Add _
    Name:=XLName, _
    RefersTo:="=" & wsParent.Name & "!" & rngNamedRange.Address

ExitPoint:
On Error Resume Next
'Must-run procedure/function clean-up code...
wbOrig.Activate
wsOrig.Activate
rngOrig.Select
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 ResizeXLNameViaRangeReferences(ByVal XLName As String, _
    ByRef RanOkRESULT As Boolean, _
    Optional ParentWkbk As Workbook, _
    Optional RANGEColNmbrForVertResizing As Long, _
    Optional RANGERowNmbrForHorizResizing As Long)
'*********
'NOTE:  This procedure is designed to return minimum dimensions of 1 row and _
1 column, even if the named region is empty.
'*********
'Error-handling declarations...
Const bSUB_IS_ENTRY_PT As Boolean = False
Const sSOURCE As String = "ResizeXLNameViaRangeReferences()"
Dim bCalledProcedureRanOk As Boolean
'Operating code declarations...
Dim wbOrig As Workbook
Dim wsOrig As Worksheet
Dim wsParent As Worksheet
Dim rngOrig As Range
Dim rngNamedRange As Range
Dim rngLastWsCell As Range
Dim rngLastColCell As Range
Dim rngLastRowCell As Range
Dim cl As Range
Dim lWsColToResizeOn As Long
Dim lWsRowToResizeOn As Long
Dim lWsRowNmbrLastCell As Long
Dim lWsColNmbrLastCell As Long
Dim lWsRowNmbrTopCell As Long
Dim lWsColNmbrTopCell As Long

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

'Set some variables...
Set wbOrig = ActiveWorkbook '...capture
Set wsOrig = ActiveSheet '...capture
Set rngOrig = Selection '...capture

'Smurf over to the relevant workbook, if it's been supplied...
If Not ParentWkbk Is Nothing Then ParentWkbk.Activate

Set rngNamedRange = Names(XLName).RefersToRange
Set rngLastWsCell = wsParent.Cells.SpecialCells(xlCellTypeLastCell)

Set cl = wsParent.Range(XLName).Cells(1, 1)
lWsRowNmbrTopCell = cl.Row
lWsColNmbrTopCell = cl.Column

'NOTE:  Using Item rather than Cells property allows us to pick rows and _
    columns beyond the range of the existing named area...
If RANGEColNmbrForVertResizing > 0 Then
    Set cl = wsParent.Range(XLName).Item(1, RANGEColNmbrForVertResizing)
    lWsColToResizeOn = cl.Column
End If
```

```
If RANGERowNmbrForHorizResizing > 0 Then
    Set cl = wsParent.Range(XLName).Item(RANGERowNmbrForHorizResizing, 1)
    lWsRowToResizeOn = cl.Row
End If

'Handle vertical resizing...
If RANGEColNmbrForVertResizing > 0 Then
    'Preliminary definitions...
    lWsRowNmbrLastCell = rngLastWsCell.Row
    Set rngLastColCell = wsParent.Cells(lWsRowNmbrLastCell, lWsColToResizeOn)
    'If the "bottom" cell is empty we have to find the true bottom cell _
    and then vertically resize/redefine the range/Name...
    If IsEmpty(rngLastColCell) Then
        wsParent.Activate
        rngLastColCell.End(xlUp).Select
        Set rngLastColCell = Selection '...TRUE bottom cell
        lWsRowNmbrLastCell = rngLastColCell.Row '...TRUE bottom row
    End If
    'Make sure we have at least one row...
    If lWsRowNmbrLastCell < lWsRowNmbrTopCell Then _
        lWsRowNmbrLastCell = lWsRowNmbrTopCell
    Set rngNamedRange = _
        rngNamedRange.Resize(1 + lWsRowNmbrLastCell - lWsRowNmbrTopCell)
End If

'Handle horizontal resizing...
If RANGERowNmbrForHorizResizing > 0 Then
    'Preliminary definitions...
    lWsColNmbrLastCell = rngLastWsCell.Column
    Set rngLastRowCell = wsParent.Cells(lWsRowToResizeOn, lWsColNmbrLastCell)
    'If the "right-most" cell is empty we have to find the true right-most _
    cell and then horizontally resize/redefine the range/Name...
    If IsEmpty(rngLastRowCell) Then
        'If we didn't activate the wsh before we must now...
        If RANGEColNmbrForVertResizing = 0 Then wsParent.Activate
        rngLastRowCell.End(xlToLeft).Select
        Set rngLastRowCell = Selection '...TRUE R-most cell
        lWsColNmbrLastCell = rngLastRowCell.Column '...TRUE R-most column
    End If
    'Make sure we have at least one column...
    If lWsColNmbrLastCell < lWsColNmbrTopCell Then _
        lWsColNmbrLastCell = lWsColNmbrTopCell
    Set rngNamedRange = _
        rngNamedRange.Resize(, 1 + lWsColNmbrLastCell - lWsColNmbrTopCell)
End If

'Now redefine the XL name...
ActiveWorkbook.Names.Add _
    Name:=XLName, _
    RefersTo:="=" & wsParent.Name & "!" & rngNamedRange.Address


ExitPoint:
On Error Resume Next
'Must-run procedure/function clean-up code...
wbOrig.Activate
wsOrig.Activate
rngOrig.Select
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
'===============================================================================
'===============================================================================
'===============================================================================
'===============================================================================
'===============================================================================
'===============================================================================
'===============================================================================
'=============================================================================
```

```
'.....................................................................................
'.....................................................................................
'.......        Written by:             William H. White (consultant)
'.......        With:                   TEKSystems
'.......                                www.teksystems.com
'.......        For:                    AIG, Inc.
'.......                                Tax Technology Group
'.......                                1 WFC, 14th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 581-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Provicence, NJ
'.......        Module created:         January 17, 2012
'.......        Proj finished:          March 21, 2012
'.....................................................................................
'.....................................................................................



'.....................................................................................
'.........................  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 AskAboutGeneralStepThruMode()
g_bStepThruMode = False
Dim ansStepThru As VbMsgBoxResult
ansStepThru = _
        MsgBox("Step-through macro?", vbYesNo, "DEVELOPMENT-ONLY MESSAGE...")
```

```
If ansStepThru = vbYes Then g_bStepThruMode = True
End Sub
'===============================================================================
Public Sub AskAboutWsChangeStepThruMode()
g_bStepThruWsChangeEventMode = False
Dim ansStepThru As VbMsgBoxResult
ansStepThru = MsgBox("Step-through Worksheet_Change event?", vbYesNo, _
    "DEVELOPMENT-ONLY MESSAGE...")
If ansStepThru = vbYes Then g_bStepThruWsChangeEventMode = True
End Sub
'===============================================================================
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

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
'================================================================================
Public Sub SetColWidths()
Dim cl As Range
Dim l As Long
For Each cl In Selection
    cl.ColumnWidth = cl
Next cl
End Sub
'================================================================================
Public Sub DupeRows()
Dim i As Byte
Dim rngSrc As Range
Set rngSrc = ActiveCell.Resize(1, 3)
For i = 1 To 80
    rngSrc.Offset(1, 0).EntireRow.Insert xlShiftDown
    rngSrc.Copy rngSrc.Offset(1, 0)
    Set rngSrc = rngSrc.Offset(2, 0)
Next i

End Sub
'================================================================================
Public Sub CopyToEveyOtherRow()
Dim i As Byte
Dim rngDest As Range
ActiveCell.Copy
Set rngDest = ActiveCell.Offset(3, 0)
For i = 1 To 78
    rngDest.PasteSpecial xlPasteAll
    Set rngDest = rngDest.Offset(3, 0)
Next i
End Sub
'================================================================================
```

vDevUtilities - 4

```
Public Sub InsertBlankRows()
Dim i As Byte
Dim rngInsRow As Range
Set rngInsRow = ActiveCell.Offset(2, 0)
For i = 1 To 80
    rngInsRow.EntireRow.Insert xlShiftDown
    Set rngInsRow = rngInsRow.Offset(2, 0)
Next i


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

MsgBoxHandler - 1

```
'.....................................................................................
'.....................................................................................
'.......        Written by:             William H. White (consultant)
'.......        With:                   TEKSystems
'.......                                www.teksystems.com
'.......        For:                    AIG, Inc.
'.......                                Tax Technology Group
'.......                                1 WFC, 14th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 581-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         January 18, 2012
'.......        Proj finished:          March 21, 2012
'.....................................................................................
'.....................................................................................



'.....................................................................................
'.............................  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!!! ***"


'**************************************
'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



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

```
MsgBoxHandler - 2

'=============================================================================
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  ===================
'============================================================

'**************************************
'SET USER MESSAGES
'**************************************
'=============================================================================
Public Sub ShowMsgBoxNoAnswer(MsgBoxStyle As VbMsgBoxStyle)
MsgBox m_sPrompt, MsgBoxStyle, m_sTitle
'Reset matters...
m_sPrompt = vbNullString
m_sTitle = vbNullString
End Sub
'=============================================================================
Public Sub SetNoSIFolderMsg()
m_sTitle = "Incomplete Data"
m_sPrompt = _
    "Please select a Discrete Excel File path before you proceed."
End Sub
'=============================================================================
Public Sub SetInvalidSIFolderMsg()
m_sTitle = "Invalid Path"
m_sPrompt = _
    "Either we cannot connect to the Discrete SI Folders File " & _
    "path you have selected, or the path shown on the spreadsheet " & _
    "is invalid." & m_s2Crs & _
    "Please use the " & _
    Chr(34) & WshDashboard.btnSelectParentFolderSIFolders.Caption & Chr(34) & _
    "  button and try re-selecting a Discrete SI Folders File Path."
End Sub
'=============================================================================
Public Sub SetInitiatingQtrMsg()
m_sTitle = "Initiating Quarter"
m_sPrompt = _
        "Initiating a quarter will:" & m_s2Crs1Tab & _
        "1) Copy all data on the two Working_Area sheets onto the " & _
        "appropriate Copy and Prior_Qtr sheets, somewhat rearranging the " & _
        "columns in the process." & m_s2Crs1Tab & _
```

```
MsgBoxHandler - 3

        "2) Clear all data from the Working_Area sheets." & m_s3Crs & _
        "Is this what you want to do?"
End Sub
'================================================================================
Public Sub SetIsQtrSetAndSIListCurrentMsg()
m_sTitle = "A Reminder..."
m_sPrompt = _
        "Have you done BOTH of the following:" & m_s2Crs1Tab & _
        "1) Set the scroll box to the CORRECT quarter, and" & m_s2Crs1Tab & _
        "2) Run the  " & _
        Chr(34) & WshDashboard.btnRetreiveSIList.Caption & Chr(34) & _
        "  button so that your list of SI sub-folders is current?"
End Sub
'================================================================================
Public Sub SetRetrievingDiscreteQtrMsg()
m_sTitle = "Retrieve Discrete Data for the Quarter?"
m_sPrompt = _
        "Proceeding with this operation will: " & m_s2Crs1Tab & _
        "1) Loop through all of the SI sub-folders." & m_s2Crs1Tab & _
        "2) Open the appropriate workbook inside each sub-folder." & _
        m_s2Crs1Tab & _
        "3) Copy and paste data from those individual workbooks into " & _
        "this workbook." & m_s3Crs & _
        "Is this what you want to do?"
End Sub
'================================================================================
Public Sub SetWkgAreaDiscrDataCopiedMsg(ByVal NoFedDataToCopy As Boolean, _
    ByVal NoStateDataToCopy As Boolean)
m_sTitle = "Quarter Initiated"

If NoFedDataToCopy And NoStateDataToCopy Then
    m_sTitle = "No Data Found"
    m_sPrompt = _
        "There was data on neither the " & WshWkgAreaFed.Name & _
        " tab/worksheet nor on the " & WshWkgAreaSt.Name & "." & m_s2Crs & _
        "Nothing has been processed."
    GoTo ExitPoint
End If

If NoFedDataToCopy Then
    m_sPrompt = _
        "Working_Area_State discrete data has been copied to the " & _
        "appropriate Copy and Prior_Qtr sheets." & m_s2Crs & _
        "There were no data on the Working_Area_Federal sheet to copy."
    GoTo ExitPoint
End If

If NoStateDataToCopy Then
    m_sPrompt = _
        "Working_Area_Federal discrete data has been copied to the " & _
        "appropriate Copy and Prior_Qtr sheets." & m_s2Crs & _
        "There were no data on the Working_Area_State sheet to copy."
    GoTo ExitPoint
End If

'If here there were data on both Federal and State Wkg Area wshs...
m_sPrompt = _
    "Working_Area discrete data has been copied to Copy and Prior_Qtr " & _
    "sheets for both State and Federal."

ExitPoint:
End Sub
'================================================================================
Public Sub SetNoSIsOnDashboardMsg()
m_sTitle = "No SI Sub-Folders"
m_sPrompt = _
    "We cannot proceed without a list of SIs on the " & WshDashboard.Name & _
    " tab." & m_s2Crs & _
    "Please browse to a valid parent of SI sub-folders and click the  " & _
    Chr(34) & WshDashboard.btnRetreiveSIList.Caption & Chr(34) & "  button."
End Sub
'================================================================================
Public Sub SetCannotFindFileMsg(ByVal FileNm As String, ByVal Path As String, _
```

```
    ByVal LogFileFullNm As String)
m_sTitle = "FILE NOT FOUND...  CONTINUE??"
m_sPrompt = _
    "We are unable to find the file:" & m_s2Crs1Tab & _
    FileNm & m_s2Crs & _
    "Within this folder:" & m_s2Crs1Tab & _
    Path & m_s3Crs & _
    "We are recording this failure in a log file:" & m_s2Crs1Tab & _
    LogFileFullNm & m_s3Crs & _
    "[NOTE:  You will be prompted about opening and viewing the log file " & _
    "when you close this application.]" & m_s2Crs & _
    "Would you like to continue retrieving quarterly data from other files?"
End Sub
'================================================================================
Public Sub SetCannotOpenFileMsg(ByVal FileFullNm As String, _
    ByVal LogFileFullNm As String)
m_sTitle = "CANNOT OPEN FILE...  CONTINUE??"
m_sPrompt = _
    "We are unable to open this file:" & m_s2Crs1Tab & _
    FileFullNm & m_s3Crs & _
    "We are recording this failure in a log file:" & m_s2Crs1Tab & _
    LogFileFullNm & m_s3Crs & _
    "[NOTE:  You will be prompted about opening and viewing the log file " & _
    "when you close this application.]" & m_s2Crs & _
    "Would you like to continue retrieving quarterly data from other files?"
End Sub
'================================================================================
Public Sub SetProblemsWorkingWithSIWbMsg(ByVal FileFullNm As String, _
    ByVal LogFileFullNm As String)
m_sTitle = "PROBLEMS WITH DISCRETE DATA FILE...  CONTINUE??"
m_sPrompt = _
    "We encountered problems with the following file:" & m_s2Crs1Tab & _
    FileFullNm & m_s2Crs & _
    "We are recording this failure in a log file:" & m_s2Crs1Tab & _
    LogFileFullNm & m_s3Crs & _
    "[NOTE:  You will be prompted about opening and viewing the log file " & _
    "when you close this application.]" & m_s2Crs & _
    "Would you like to continue retrieving quarterly data from other files?"
End Sub
'================================================================================
Public Sub SetSIWbDiscrWshCorruptedMsg(ByVal FileFullNm As String, _
    ByVal DiscreteWshNm As String, ByVal LogFileFullNm As String)
m_sTitle = "DISCRETE TAB/SHEET CORRUPTED...  CONTINUE??"
m_sPrompt = _
    "The two Discrete tabs in the SI Excel Workbooks are supposed to have " & _
    "EXACTLY " & g_ySIWB_DISCRWS_COLS & " columns.  However, it appears " & _
    "that the following worksheet has been corrupted:" & m_s2Crs1Tab & _
    "Excel file:" & vbTab & FileFullNm & m_s2Crs1Tab & _
    "Worksheet/tab:" & vbTab & DiscreteWshNm & m_s3Crs & _
    "We are recording this failure here:" & m_s2Crs1Tab & _
    "Log file:" & vbTab & LogFileFullNm & m_s3Crs & _
    "Further, when you are done running this application we will " & _
    "automatically open this log file for you (in Notepad) on your " & _
    "screen." & m_s2Crs & _
    "Would you like to continue retrieving quarterly data from other files?"
End Sub
'================================================================================
Public Sub SetDiscrDataPopulatedMsg(ByVal YrQtr As String)
m_sTitle = "Data Retrieved"
m_sPrompt = _
    YrQtr & " Discrete data have been populated in the Working Area " & _
    "sheets for both Federal and State."
End Sub
'================================================================================
Public Sub SetThereWereProblemFilesMsg()
m_sTitle = "Reminder..."
m_sPrompt = _
    "It appears that this application experienced problems with one or more " & _
    "of the Discrete Data files." & m_s2Crs & _
    "Do you want to see the associated log files now?"
End Sub
'================================================================================
Public Sub SetCannotOpenProbFilesLogMsg(ByVal LogFileFullNm As String)
```

```
MsgBoxHandler - 5

m_sTitle = "Problem Opening Log File"
m_sPrompt = _
    "Sorry.  We are unable to open the log file:" & m_s2Crs1Tab & _
    LogFileFullNm
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 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 & _
        "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


'**************************************
'SET 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?"
```

```
MsgBoxHandler - 6

End Sub

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