ThisWorkbook - 1

```
Option Explicit
'================================================================================
'================================================================================
Private Sub Workbook_Open()
Dim bStepThruMode As Boolean
Set g_cMBxHndlr = New MsgBoxHandler

'Just to cover ourselves for when we are firing this event 'by hand' _
during development...
g_bExecuteWbkOnCloseCode = False
g_bDebugMode = False
bStepThruMode = False

'Capture original status (we'll restore on exit)...
g_ffOrigFileSaveMode = Application.DefaultSaveFormat
Application.DefaultSaveFormat = xlOpenXMLWorkbookMacroEnabled

'Handle debug mode...
If ThisWorkbook.CustomDocumentProperties("Production Mode").Value Then
    g_enExecMode = projProdActual
Else
    'Prompt for running in debug mode...
    Dim ansRunInDebugMode As VbMsgBoxResult
    g_cMBxHndlr.vSetRunDebugModeQues
    ansRunInDebugMode = g_cMBxHndlr.ShowMsgBoxReturnAnswer(vbYesNo)
    If ansRunInDebugMode = vbNo Then
        g_enExecMode = projProdTest
    Else
        'Can't set g_enExecMode yet...
        g_bDebugMode = True
    End If
End If

'Handle step-thru mode...
If g_bDebugMode Then
    Dim ansRunInStepThruMode As VbMsgBoxResult
    g_cMBxHndlr.vSetStepThruModeQues
    ansRunInStepThruMode = g_cMBxHndlr.ShowMsgBoxReturnAnswer(vbYesNo)
    If ansRunInStepThruMode = vbNo Then
        g_enExecMode = projDebugNoStops
    Else
        g_enExecMode = projDebugStepThru
        bStepThruMode = True
    End If
End If

'More debugging questions...
If g_enExecMode <> projProdActual Then
    Dim ansBeAReviewer As VbMsgBoxResult
    Dim ansBeASuperRvwr As VbMsgBoxResult
    Dim ansIgnoreCannotPrepOthersWbk As VbMsgBoxResult

    g_cMBxHndlr.vSetBeAReviewerQues
    ansBeAReviewer = g_cMBxHndlr.ShowMsgBoxReturnAnswer(vbYesNo)
    If ansBeAReviewer = vbYes Then g_bIAmReviewer = True

    g_cMBxHndlr.vSetBeASuperReviewerQues
    ansBeASuperRvwr = g_cMBxHndlr.ShowMsgBoxReturnAnswer(vbYesNo)
    If ansBeASuperRvwr = vbYes Then g_bIAmSuperReviewer = True

    g_cMBxHndlr.vSetIgnoreDiffPreparerTrapQues
    ansIgnoreCannotPrepOthersWbk = g_cMBxHndlr.ShowMsgBoxReturnAnswer(vbYesNo)
    If ansIgnoreCannotPrepOthersWbk = vbYes Then _
        g_bSkipTrapForPreppingSomeoneElsesWork = True

End If

If bStepThruMode Then Stop

Application.DisplayDocumentInformationPanel = True
```

```
ThisWorkbook - 2

'Now fire up the workbook...
Main.ProtectPreparerAndReviewerWshs
Main.SetVisibilityOnAllButInstrucWshs
Main.VerifyCTXAndXL07AndPrepInstrWsh

'See if this is an approved wbk..
If Range("ptrFREDRvwrStatusID").Value = g_dRVW_STATUS_ID_APPRVD _
    Then g_bReviewerApproved = True
Main.SetWbkApprovedMsgVisibilityOnInstrucWsh MsgVisible:=g_bReviewerApproved

''See if user remembered to check-out the wbk from ShPt...
'If ThisWorkbook.CanCheckIn Then
'    'We use this global to restrict access to the workbook...
'    g_bWbCheckedOutSP = True
'Else
'    g_cMBxHndlr.SetWkbkNotChkdOutMsg
'    g_cMBxHndlr.ShowMsgBoxNoAnswer vbInformation
'End If

Range("Selected_Role").Select
Range("Selected_Role").Activate
End Sub
'================================================================================
Private Sub Workbook_BeforeClose(Cancel As Boolean)
'If Not ThisWorkbook.CustomDocumentProperties("Production Mode").Value Then Stop
If Not g_bExecuteWbkOnCloseCode Then Exit Sub

'No error-trapping to be done here...
On Error Resume Next

Application.ScreenUpdating = False
WshInstruction.Select

'Set info cells to indicate worst-case scenarios.  These will be changed, _
if appropriate, when the wbk is re-opened...
WshInstruction.Protect Password:=g_sWSH_PWD, userinterfaceonly:=True
WshInstruction.Range("ptrXLVersion").ClearContents
WshInstruction.Range("ptrCTXInstalled").ClearContents
WshInstruction.Range("ptrMacrosEnabled") = "No"
WshInstruction.Range("CTX_Enterprise").ClearContents

'Disable macro button...
WshInstruction.btnUnhideAndPrepRoleBasedWsh.Enabled = False

'Hide all but Instruction worksheet...
WshReviewer.Visible = xlSheetVeryHidden
WshPreparer.Visible = xlSheetVeryHidden

Application.ScreenUpdating = True

'Restore matters...
On Error Resume Next '...just in case the variable somehow is befouled
Application.DefaultSaveFormat = g_ffOrigFileSaveMode

End Sub
'================================================================================
Private Sub Workbook_BeforeSave(ByVal SaveAsUI As Boolean, Cancel As Boolean)
If g_enExecMode = projDebugStepThru Then Stop

'To handle development environment...
If g_cMBxHndlr Is Nothing Then Set g_cMBxHndlr = New MsgBoxHandler

If g_bInvalidEntCode Or g_bEntCodeNotOnSP Then
    g_cMBxHndlr.SetSaveDisabledDueToInvalidCTXCodeMsg
    g_cMBxHndlr.ShowMsgBoxNoAnswer vbExclamation
    Cancel = True
End If

'NOTE:  we trapped for invalid year entries in-process.  If SP does not _
agree with Wsh go ahead and change SP
If Not g_bEntCodeNotOnSP Then
```

```
    SPMetaDataRelated.SynchShPtPropertiesWithXLEntries
Else '...our global disallow-save boolean = true
    Cancel = True
End If

ExitPoint:
    If g_enExecMode = projDebugNoStops Or _
            g_enExecMode = projDebugStepThru Then Cancel = False
    Exit Sub

End Sub
```

WshInstruction - 1

```
Option Explicit
'================================================================================
Private Sub btnUnhideAndPrepRoleBasedWsh_Click()
BtnHandlers.DashboardUnhideAndPrepRoleBasedWsh
End Sub
'================================================================================
Private Sub btnPrintRvwrWshAllRows_Click()
BtnHandlers.DashboardPrintPreviewRvwrWsh AllRows:=True
End Sub
'================================================================================
Private Sub btnPrintRvwrWshExcepRows_Click()
BtnHandlers.DashboardPrintPreviewRvwrWsh AllRows:=False
End Sub
'================================================================================
```

WshPreparer - 1

```
Option Explicit
Private m_rngEntAndPerCells As Range
Private m_bWeChangedTaxYear As Boolean '...FALSE
Private m_bSkipEOPCheck As Boolean '...FALSE
'================================================================================
Private Sub btnUnlockInputRng_Click()
If g_enExecMode = projDebugStepThru Then Stop
BtnHandlers.PreparerUnlockInputRange
End Sub
'================================================================================
Private Sub btnSubmitData_Click()
If g_enExecMode = projDebugStepThru Then Stop
BtnHandlers.PreparerSubmitData
End Sub
'================================================================================
Private Sub btnExpndCollapsePrep_Click()
BtnHandlers.PrepRvwrExpandCollapseFrozenPaneRows projPreparer
End Sub
'================================================================================
Private Sub Worksheet_Change(ByVal Target As Range)
Const sBAD_ENTITY_FLAG As String = "#InvalidEntity"
Const sSUBMITTED_STATUS As String = "Submitted for Review"

'Bail if we are shutting down "by hand"...
If g_bDevShutDownMode Then Exit Sub

'Bail if we are not in our cells of interest...
If Application.Intersect(Target, WshPreparer.Range("rgnWshChangeCells")) _
    Is Nothing Then Exit Sub

'Define a range where we can fire a worksheet change if the user changes _
either the Entity Code or Period entry in the worksheet...
Set m_rngEntAndPerCells = _
    Application.Union(Range("CTX_Ent_Code"), Range("Period"))

'*********************************************************
'*********************************************************
'TO STEP THROUGH THE WSH_CHANGE EVENT ADD A WATCH FOR:
'   Not Application.Intersect(Target,Range("CTX_Ent_Code")) is Nothing
'   Not Application.Intersect(Target,Range("Period")) is Nothing
'   Not Application.Intersect(Target,Range("Status_Prep")) is Nothing
'AND SET WATCH TO BREAK-WHEN-TRUE
'*********************************************************
'*********************************************************

'Handle changes to the Preparer status...
If Not Application.Intersect(Range("Status_Prep"), Target) Is Nothing And _
    Not g_bReviewerRejectedFile And _
    StrComp(UCase(Target), UCase(sSUBMITTED_STATUS)) = 0 Then _
        BtnHandlers.PreparerSubmitData

'Handle changes to the tax year...
If Not Application.Intersect(Range("Period"), Target) Is Nothing Then
    If Not m_bWeChangedTaxYear Then
        Dim bTaxYearValid As Boolean
        bTaxYearValid = _
            SPMetaDataRelated.IsXLEntryValidShPtEntry_Simple( _
                XLWshEntry:=Trim(Target), _
                SPPptyName:=g_sSP_PPTY_TAX_YR)
        'Set flag if we are going to have to change the tax year...
        If Not bTaxYearValid Then
            SPMetaDataRelated.NotifyUserOfIncorrectTaxYearSelection
            'Changing the tax year to "last year" [NOTE:  Making this change _
            immediately refires the Worksheet_Change event handler.  The event _
            handler will run all the way through once and then restart right _
            here to finish handling the original change.  We created and set the _
            module-level boolean m_bWeChangedTaxYear so that we can catch that we _
            will have to double-check/change the EOP year.]...
            m_bWeChangedTaxYear = True '...we must set this one line early
            Target = Year(Now()) - 1
        End If
```

```
        'If we have changed the tax year we must also change EOP...
        m_bSkipEOPCheck = True '...since we know we have a valid year
        SPMetaDataRelated.UpdateEndOfPeriodForNewTaxYear
        'Set the SP property...
        ThisWorkbook.ContentTypeProperties(g_sSP_PPTY_TAX_YR).Value = _
            Trim(Range("Period"))
    Else
        'm_bWeChangedTaxYear = False '...reset flag
    End If
End If

'Handle changes to Period End date...
If Not Application.Intersect(Range("Period_End"), Target) Is Nothing Then
    If Not m_bSkipEOPCheck Then
        If g_enExecMode = projDebugStepThru Then Stop
        Main.EnsureValidPeriodEnd
    Else
        'Turn switch back off...
        m_bSkipEOPCheck = False
    End If
    'Update/set the SP Pppt...
    ThisWorkbook.ContentTypeProperties(g_sSP_PPTY_EOP).Value = _
            Range("Period_End").Value
End If

'Handle updates to Entity Code...
If Not Application.Intersect(Range("CTX_Ent_Code"), Target) Is Nothing Then
    'Trap for an invalid entity code...
    Dim bWbSaveAlreadyDisabled As Boolean
    Main.InvokeCTXCalc projWsh
    If g_enExecMode = projDebugStepThru Then Stop
    If Range("FRED_Profile") = sBAD_ENTITY_FLAG Then
        g_cMBxHndlr.SetBadEntityCodeMsg
        g_cMBxHndlr.ShowMsgBoxNoAnswer vbInformation
        g_bInvalidEntCode = True
        SPMetaDataRelated.ClearEnumeratedShPtPpty _
            SPPpty:=g_sSP_PPTY_ENT_SNM
        SPMetaDataRelated.ClearEnumeratedShPtPpty _
            SPPpty:=g_sSP_PPTY_SICODE_SINM
        ThisWorkbook.BuiltinDocumentProperties("Title") = vbNullString
        GoTo ExitPoint
    Else
        'In case user is correcting a prior bad entry...
        g_bInvalidEntCode = False
    End If

    'If here entity code is valid.  However, we NOW need to trap for _
    not being able to find the entity code on SP...
    Dim bFoundCTXEntityInSPMetaData As Boolean
    bFoundCTXEntityInSPMetaData = _
        SPMetaDataRelated.IsXLEntryInSPEnumValsLeftMostChars( _
            XLEntry:=Target, _
            SPPpty:=g_sSP_PPTY_ENT_SNM)
    If bFoundCTXEntityInSPMetaData Then
        g_bEntCodeNotOnSP = False
    Else
        g_bEntCodeNotOnSP = True
        g_cMBxHndlr.SetEntCodeNotInSPMsg
        g_cMBxHndlr.ShowMsgBoxNoAnswer vbInformation
        SPMetaDataRelated.ClearEnumeratedShPtPpty _
            SPPpty:=g_sSP_PPTY_ENT_SNM
        SPMetaDataRelated.ClearEnumeratedShPtPpty _
            SPPpty:=g_sSP_PPTY_SICODE_SINM
        ThisWorkbook.BuiltinDocumentProperties("Title") = vbNullString
        'Disable most functionality of the worksheet...
        SetLockedStatusAndColorForXLRng _
            XLRangeName:="Input_Prep", _
            LockRange:=True
        SetLockedStatusAndColorForXLRng _
            XLRangeName:="CTX_Ent_Code", _
            LockRange:=False
```

```
    End If
    'Set enabled status of Submit Data butoon...
    WshPreparer.btnSubmitData.Enabled = Not g_bEntCodeNotOnSP
    If g_bEntCodeNotOnSP Then GoTo ExitPoint

    'Trap to make sure that SI code is on SP...
    Dim sEmbeddedSICode As String
    Dim bFoundSICodeInSPMetaData As Boolean

    sEmbeddedSICode = _
        Mid(Target, g_yBEG_POS_SICODE_IN_CTXCODE, g_yLEN_SICODE)
    bFoundSICodeInSPMetaData = _
        SPMetaDataRelated.IsXLEntryInSPEnumValsLeftMostChars( _
            XLEntry:=sEmbeddedSICode, _
            SPPpty:=g_sSP_PPTY_SICODE_SINM)

    If bFoundSICodeInSPMetaData Then
        'Set two SP properties...
        ThisWorkbook.ContentTypeProperties(g_sSP_PPTY_ENT_SNM).Value = _
            SPMetaDataRelated.GetFullSPPptyValueFromLHMostChars( _
                LHMostChars:=Target, _
                SPPpty:=g_sSP_PPTY_ENT_SNM)
        ThisWorkbook.ContentTypeProperties(g_sSP_PPTY_SICODE_SINM).Value = _
            SPMetaDataRelated.GetFullSPPptyValueFromLHMostChars( _
                LHMostChars:=sEmbeddedSICode, _
                SPPpty:=g_sSP_PPTY_SICODE_SINM)
    Else
        g_cMBxHndlr.SetSICodeNotFoundInSPMsg
        g_cMBxHndlr.ShowMsgBoxNoAnswer vbInformation
        SPMetaDataRelated.ClearEnumeratedShPtPpty _
            SPPpty:=g_sSP_PPTY_SICODE_SINM
        GoTo ExitPoint
    End If
End If

'Update Doc title if entity or year are changed...
If Not Application.Intersect(m_rngEntAndPerCells, Target) Is Nothing Then
    'Using this flag prevents going through this code twice...
    If Not m_bWeChangedTaxYear Then
        Main.InvokeCTXCalc projWsh
        If g_enExecMode = projDebugStepThru Then Stop
        SPMetaDataRelated.SetDocTitle

        'Save the file if we have a valid entity code, valid SI number, _
        and the entity code <> "DC" (Period is ALWAYS valid)
        '[NOTE:  chr(95) = underscore ("_"), chr(47) = fwd slash ("/")]...
        If Not g_bInvalidEntCode And Not g_bEntCodeNotOnSP And _
            StrComp(Range("CTX_Ent_Code"), g_sEMPTY_CTX_PLUG) <> 0 Then
                Dim sFlNm As String
                'multi-step build...
                sFlNm = _
                    ThisWorkbook.ContentTypeProperties("URL").Value
                If StrComp(Right(sFlNm, 1), Chr(47)) <> 0 Then _
                    sFlNm = sFlNm & Chr(47)
                sFlNm = sFlNm & Trim(Range("Period")) & Chr(95) & _
                    Range("CTX_Ent_Code") & Chr(95) & g_sTITLE_SUFFIX & _
                    g_sXL_MACRO_FILE_EXT
                ThisWorkbook.SaveAs sFlNm, _
                    XlFileFormat.xlOpenXMLWorkbookMacroEnabled
        End If
    Else
        m_bWeChangedTaxYear = False '...reset flag
    End If
End If

'Handle changes to Preparer and/or Preparer Status...
If Not Application.Intersect(Range("Prep_nm"), Target) Is Nothing Then _
    ThisWorkbook.ContentTypeProperties(g_sSP_PPTY_PREP_NM).Value = _
        Range("Prep_nm").Value
If Not Application.Intersect(Range("Status_Prep"), Target) Is Nothing Then _
    ThisWorkbook.ContentTypeProperties(g_sSp_PPTY_STATUS_PREP).Value = _
```

```
WshPreparer - 4

        Range("Status_Prep").Value


ExitPoint:
    Exit Sub
End Sub
'================================================================================
```

WshReviewer - 1

```
Option Explicit
Private m_bUserUndoingChange As Boolean
'================================================================================
Private Sub btnRefreshCTXData_Click()
BtnHandlers.ReviewerRefreshData
End Sub
'================================================================================
Private Sub btnSubmitRvwrStatus_Click()
BtnHandlers.ReviewerUpdateStatus
End Sub
'================================================================================
Private Sub btnExpndCollapseRvwr_Click()
BtnHandlers.PrepRvwrExpandCollapseFrozenPaneRows projReviewer
End Sub
'================================================================================
Private Sub Worksheet_Change(ByVal Target As Range)
Const sREJ As String = "Reject"
Const sIN_PROG As String = "In Progress"

'Bail if we are shutting down "by hand" or user is undoing a change _
[specifically, after our prompt, us has thought better of removing his _
Approval as a Reviewer]...
If g_bDevShutDownMode Or m_bUserUndoingChange Then Exit Sub

'Synch SP properties with changes to Reviewer Name...
If Not Application.Intersect(Range("Reviewer_nm"), Target) Is Nothing Then _
    ThisWorkbook.ContentTypeProperties(g_sSP_PPTY_REVIEW_NM).Value = _
        Range("Reviewer_nm").Value

'Handle changes to reviewer status...
If Not Application.Intersect(Range("Status_Review"), Target) Is Nothing Then
    ThisWorkbook.ContentTypeProperties(g_sSP_PPTY_STATUS_REVIEW).Value = _
        Range("Status_review").Value
    'Handle a reviewer rejecting a preparer's submission...
    If StrComp(sREJ, Target.Value) = 0 Then
        Dim visOrig As XlSheetVisibility
        visOrig = WshPreparer.Visible '...capture status
        If g_enExecMode = projProdActual Or g_enExecMode = projProdTest Then _
            Application.ScreenUpdating = False
        WshPreparer.Visible = xlSheetVisible
        WshPreparer.Select
        WshPreparer.Unprotect g_sWSH_PWD
        'We are about to change Reviewer Status on the Preparer wsh. _
        Normally a change to this cell fires a submit CTX data call.  We ALSO _
        will want to submit CTX data for the preparer wsh.  The CTX Submit Data _
        method generates a pop-up.  Unless we "do something" the user will _
        receive two, consecutive such CTX pop-ups.  Not only will this be _
        somewhat annoying, the risk is that the user might well think the _
        second CTX prompt is the supuerflous result of a promgramming error. _
        In that case the user is apt to answer NO to the second prompt, which _
        would create obvious problems.  We thus want to disable the _
        Submit CTX Data call normally fired when the Reviewer Status cell _
        changes on the Preparer worksheet.  We accomplish that little goal _
        via a global boolean flag...
        g_bReviewerRejectedFile = True
        WshPreparer.Range("Status_Prep") = sIN_PROG
        WshPreparer.Range("Reviewer_Status").Calculate
        WshPreparer.Protect _
            Password:=g_sWSH_PWD, _
            userinterfaceonly:=True, _
            AllowFormattingColumns:=True, _
            AllowFormattingRows:=True, _
            AllowSorting:=True, _
            AllowFiltering:=True
        WshReviewer.Select
        WshPreparer.Visible = visOrig '...restore status
        g_bReviewerRejectedFile = False '...restore to default
        Application.ScreenUpdating = True
    End If
    'Handle a reviewer changing the Review Status FROM "Approved"...
    If g_bReviewerApproved Then
```

WshReviewer - 2

```
        Dim ansReverseApproval As VbMsgBoxResult
        g_cMBxHndlr.SetReviewerUNApproveQues
        ansReverseApproval = g_cMBxHndlr.ShowMsgBoxReturnAnswer(vbYesNo)
        If ansReverseApproval = vbYes Then
            g_bReviewerApproved = False '...switch global boolean
            WshReviewer.btnSubmitRvwrStatus.Enabled = True
        Else '...user said leave as "Approved"!!!!
             m_bUserUndoingChange = True
             Target = g_sRVW_STATUS_APPVD
             GoTo ExitPoint
        End If
    End If

    If Not m_bUserUndoingChange Then
        g_objCTXOffice.SubmitDataWorkbook
        Main.InvokeCTXCalc projWkbk
    End If
End If

ExitPoint:
    m_bUserUndoingChange = False '...reset
    Exit Sub

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


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

WshSprRvwrsLnk - 1

```
Option Explicit
```

```
BtnHandlers - 1

Option Explicit
'================================================================================
Public Sub DashboardUnhideAndPrepRoleBasedWsh()
Dim visWshPreparerOrig As XlSheetVisibility
If g_enExecMode = projDebugStepThru Then Stop

Main.InvokeCTXCalc projWsh '...dashboard
If g_enExecMode = projDebugStepThru Then Stop

''To handle cases where user opened a workbook without checking it out of SP, _
'saw our MsbBox prompt, and then clicked on the EditWorkbook button (which _
'Checks Out the wb)...
'If Not g_bWbCheckedOutSP Then
'    If ThisWorkbook.CanCheckIn Then g_bWbCheckedOutSP = True
'End If

'Trap for user not having a profile set up...
If Left(Range("CTX_Enterprise").Value, 1) = "#" Then
    g_cMBxHndlr.SetNotLoggedIntoCTXMsg
    g_cMBxHndlr.ShowMsgBoxNoAnswer vbInformation
    GoTo ExitPoint
End If

'Refresh the preparer worksheet...
If g_enExecMode = projProdActual Then Application.ScreenUpdating = False
visWshPreparerOrig = WshPreparer.Visible '...capture current state
WshPreparer.Visible = xlSheetVisible
WshPreparer.Select
Main.InvokeCTXCalc projWsh
If g_enExecMode = projDebugStepThru Then Stop
WshInstruction.Select
WshPreparer.Visible = visWshPreparerOrig '...restore matters

If g_enExecMode = projDebugStepThru Then Stop

'If here then the user IS logged in and we can handle the button click...
Dim enumRole As ProjRole

If Range("Selected_Role") = "Reviewer" Then
    enumRole = projReviewer
Else
    enumRole = projPreparer
End If

If enumRole = projReviewer Then
    'Trap for whether user belongs to Fred_reviewer AD group [create _
    otherwise-unneeded boolean variable so that we can bypass this check/trap _
    during development]...
    Dim bUserIsReview As Boolean
    bUserIsReview = Main.IsMemberFREDReviewerADGrp()
    If g_bIAmReviewer Then bUserIsReview = True
    If Not bUserIsReview Then
        g_cMBxHndlr.SetNotAReviewerMsg
        g_cMBxHndlr.ShowMsgBoxNoAnswer vbExclamation
        GoTo ExitPoint
    Else '...user is a legit reviewer
        Main.DisplayRoleBasedWsh projReviewer
        Main.InitializeReviewerWsh
    End If
Else '...projPreparer
    Main.DisplayRoleBasedWsh projPreparer
    Main.InitializePreparerWsh
End If

ExitPoint:
    Application.ScreenUpdating = True
    Exit Sub

End Sub
'================================================================================
Public Sub PreparerUnlockInputRange()
```

```
BtnHandlers - 2

Const sBAD_CTX_ID As String = "#InvalidDataProfileId"
Const sCTX_INVAL_MSG_BEG As String = "#Invalid"
'NOTE:  I HATE duplicate code, and I TRIED to condense the 3 invocations of _
Main.SetLockedStatusAndColorForXLRng into a single call (passing in variables/ _
parameters for the range name and locked status), but it got tricky and _
I felt it would ultimately be a counter-productive exercise... - WHW

'Just in case user has de-activated auto-recalc...
Range("Reviewer_Status").Calculate

'Trap for the file being in review...
If Main.IsFileInReview() Then
    Main.SetLockedStatusAndColorForXLRng _
        XLRangeName:="Input_Prep", _
        LockRange:=True
    g_cMBxHndlr.SetAlreadyInReviewMsg
    g_cMBxHndlr.ShowMsgBoxNoAnswer vbInformation
    WshPreparer.btnUnlockInputRng.Enabled = False
Else '...NOT currently in review
    'Trap for an invalid CTX code...
    If StrComp(Left(Range("FRED_Profile"), Len(sCTX_INVAL_MSG_BEG)), _
            sCTX_INVAL_MSG_BEG) = 0 Then
        g_cMBxHndlr.SetBadEntityCodeMsg
        g_cMBxHndlr.ShowMsgBoxNoAnswer vbInformation
        Main.SetLockedStatusAndColorForXLRng _
            XLRangeName:="CTX_Ent_Code", _
            LockRange:=False
        GoTo ExitPoint
    End If
    'Trap for CTX plug entry...
    If StrComp(Trim(Range("CTX_Ent_Code")), g_sEMPTY_CTX_PLUG) = 0 Then
        Main.SetLockedStatusAndColorForXLRng _
            XLRangeName:="CTX_Ent_Code", _
            LockRange:=False
        GoTo ExitPoint
    End If

    'If here we have a valid CTX code, so unlock range...
    Main.SetLockedStatusAndColorForXLRng _
        XLRangeName:="Input_Prep", _
        LockRange:=False
    Range("Input_Start").Select
End If

ExitPoint:
    Exit Sub

End Sub
'================================================================================
Public Sub PreparerSubmitData()

'Trap for an already-approved workbook...
If Range("Reviewer_Status") = "Approved" Then
    g_cMBxHndlr.SetCannotSubmitPostApprovalMsg
    g_cMBxHndlr.ShowMsgBoxNoAnswer vbInformation
    WshPreparer.btnSubmitData.Enabled = False
    GoTo ExitPoint
End If

'Test for existence/validity of object...
If g_objCTXOffice Is Nothing Then
    Dim bAttemptSuccessful As Boolean
    Main.RecreateCTXOffObj RecreatedRESULT:=bAttemptSuccessful
    If Not bAttemptSuccessful Then GoTo InvalidCTXObj
End If

'CTX itself will generate an "Are-You-Sure?" pop-up.  [The original code _
for this VBAProj included the follwing line of code: _
    g_objCTXOffice.SubmitDataMap g_sFRED_INPUT_ID _
However, in testing WHW & HFC found this line didn't always accomplish what _
we wanted it to.  The code has thus been changed to something that _
```

```
BtnHandlers - 3

seems to work much more reliably.]...
g_objCTXOffice.SubmitDataWorksheet

Main.InvokeCTXCalc projWsh
If g_enExecMode = projDebugStepThru Then Stop

ExitPoint:
    Exit Sub

InvalidCTXObj:
    g_cMBxHndlr.SetCannotConnectToCTXOfficeMsg
    g_cMBxHndlr.ShowMsgBoxNoAnswer vbExclamation
    GoTo ExitPoint
End Sub
'================================================================================
Public Sub ReviewerUpdateStatus()

'Test for existence/validity of object...
If g_objCTXOffice Is Nothing Then
    Dim bAttemptSuccessful As Boolean
    Main.RecreateCTXOffObj RecreatedRESULT:=bAttemptSuccessful
    If Not bAttemptSuccessful Then GoTo InvalidCTXObj
End If

g_objCTXOffice.SubmitDataWorksheet

Main.InvokeCTXCalc projWsh

'Handle reviewer Approval (making sure that the reviewer said "Yes" to _
submitting the data to CTX)...
If Range("CTX_Status_Review") = Range("ptrFREDRvwrStatusID") = _
        g_dRVW_STATUS_ID_APPRVD Then
    g_bReviewerApproved = True
    Main.SetLockedStatusAndColorForXLRng _
        XLRangeName:="ptrVisOrgChartVfd", _
        LockRange:=True
    WshReviewer.btnSubmitRvwrStatus.Enabled = False
End If

ExitPoint:
    Exit Sub

InvalidCTXObj:
    g_cMBxHndlr.SetCannotConnectToCTXOfficeMsg
    g_cMBxHndlr.ShowMsgBoxNoAnswer vbExclamation
    GoTo ExitPoint

End Sub
'================================================================================
Public Sub ReviewerRefreshData()
Main.InvokeCTXCalc projWsh
If g_enExecMode = projDebugStepThru Then Stop
End Sub
'================================================================================
Public Sub DashboardPrintPreviewRvwrWsh(ByVal AllRows As Boolean)
Const sPG_HDR_FULL As String = "Reviewer - All Data"
Const sPG_HDR_EXCEP As String = "Reviewer - Exception Data ONLY"

Dim visWshRvwrOrig As XlSheetVisibility

If g_enExecMode = projDebugStepThru Then Stop
Main.InvokeCTXCalc projWkbk
If g_enExecMode = projDebugStepThru Then Stop

'This button, unlike the "Continue" button, seems to require a second recalc...
Main.InvokeCTXCalc projWsh
If g_enExecMode = projDebugStepThru Then Stop

'Trap for user not having a profile set up...
If Left(Range("CTX_Enterprise").Value, 1) = "#" Then
    g_cMBxHndlr.SetNotLoggedIntoCTXMsg
```

```
BtnHandlers - 4

    g_cMBxHndlr.ShowMsgBoxNoAnswer vbInformation
    GoTo ExitPoint
End If

If g_enExecMode = projProdActual Then Application.ScreenUpdating = False

visWshRvwrOrig = WshReviewer.Visible '...capture current state
WshReviewer.Visible = xlSheetVisible
WshReviewer.Select

If Not AllRows Then
    Dim rngDataToFilter As Range
    Dim lDiffCol As Long
    Dim lColsInRng As Long
    Dim i As Long

    Set rngDataToFilter = WshReviewer.Range("rgnwhDataToFilter_Rvwr")
    lColsInRng = rngDataToFilter.Columns.Count

    'Find column number WITHIN RANGE of cols to filter on...
    For i = 1 To lColsInRng
        If Not Application.Intersect(rngDataToFilter.Columns(i), _
            WshReviewer.Range("colDiff")) Is Nothing Then
                lDiffCol = i
            i = lColsInRng '...end loop
        End If
    Next i
    If g_enExecMode = projDebugStepThru Then Stop

    'Filter...
    rngDataToFilter.AutoFilter _
        Field:=lDiffCol, _
        Criteria1:="<>0", _
        visibledropdown:=True

    'For some reason filtering causes our row outline to expand, _
    so re-collapse them...
    WshReviewer.Outline.ShowLevels 1, 1
    Main.InvokeCTXCalc projWkbk
    If g_enExecMode = projDebugStepThru Then Stop
    WshReviewer.PageSetup.CenterHeader = sPG_HDR_EXCEP
Else
    WshReviewer.PageSetup.CenterHeader = sPG_HDR_FULL
End If

WshReviewer.PrintPreview EnableChanges:=True
If g_enExecMode = projDebugStepThru Then Stop

'User has now closed out of PrintPreview...
If Not AllRows Then
    WshReviewer.AutoFilterMode = False
    'Again, a change in filters causes our row outline to expand...
    WshReviewer.Outline.ShowLevels 1, 1
    'A change in outline levels also causes CTX formulas to barf...
    Main.InvokeCTXCalc projWkbk
    If g_enExecMode = projDebugStepThru Then Stop
End If

WshInstruction.Select
WshReviewer.Visible = visWshRvwrOrig '...restore matters

ExitPoint:
    Application.ScreenUpdating = True
    Exit Sub

End Sub
'================================================================================
Public Sub PrepRvwrExpandCollapseFrozenPaneRows(ByVal WhichSheet As ProjRole)
Const sCPTN_EXP As String = "Expand Top Pane"
Const sCPTN_COLL As String = "Collapse Top Pane"
```

```
BtnHandlers - 5

Dim sWsNm As String
Dim sNewBtnCaption As String
Dim bRowsCurrentlyHidden As Boolean '...default = false

Select Case g_enExecMode
    Case projDebugStepThru
        Stop
    Case projProdActual
        Application.ScreenUpdating = False
End Select

Select Case WhichSheet
    Case projPreparer
        sWsNm = WshPreparer.Name
    Case projReviewer
        sWsNm = WshReviewer.Name
End Select

If Worksheets(sWsNm).Range("rowCase").EntireRow.Hidden Then
    bRowsCurrentlyHidden = True
    sNewBtnCaption = sCPTN_COLL
Else '...bRowsCurrentlyHidden already FALSE
    sNewBtnCaption = sCPTN_EXP
End If

'Toggle hidden status of our rows...
Worksheets(sWsNm).Range("rowCase").EntireRow.Hidden = _
    Not bRowsCurrentlyHidden
Worksheets(sWsNm).Range("rowsIDInfoToHide").EntireRow.Hidden = _
    Not bRowsCurrentlyHidden

'Toggle button caption, and visibility of other buttons...
Select Case WhichSheet
    Case projPreparer
        With WshPreparer
            .btnExpndCollapsePrep.Caption = sNewBtnCaption
            .btnUnlockInputRng.Visible = bRowsCurrentlyHidden
            .btnSubmitData.Visible = bRowsCurrentlyHidden
        End With
    Case projReviewer
        With WshReviewer
            .btnExpndCollapseRvwr.Caption = sNewBtnCaption
            .btnRefreshCTXData.Visible = bRowsCurrentlyHidden
            .btnSubmitRvwrStatus.Visible = bRowsCurrentlyHidden
        End With
End Select

'Hiding/unhiding rows tends to send all CTX formulas into paroxysms, so...
Main.InvokeCTXCalc projWsh
If g_enExecMode = projDebugStepThru Then Stop

Application.ScreenUpdating = True

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

```
Option Explicit

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

Public Const g_sAPP_TITLE As String = "FRED Template"
Public Const g_sWSH_PWD As String = "nochange"

'**************************************
'**************************************
'NOTE: THE FOLLOWING COMMENT is left in case it is every necessary _
to revert to utilizing the SubmitDataMap() procedure.  As commented in the _
BtnHandlers.PreparerSubmitData() procedure, we have discontinued _
using SubmitDataMap() and now use SubmitDataWorksheet():

''To submit data from within VBA you run the SubmitDataMap() procedure on the _
'CTXOffice object, passing in an argument which is the XML id of your data map.
''The XML you need is found in the WshDataCache (i.e., %CORPTAX_DATA_CACHE%) _
'sheet.  Generally you will have to click on the cells at the top of column A _
'to read the XML.
''The XML element you want will read (roughly) as follows:
''<DataMap name="FRED_Input" description="whatever..." id="DataMap_1" _
'    rangeAddress="whatever" etc... />
''In this case if you want to submit under "FRED_Input" you would pass in _
'"DataMap_1" as you argument in the SubmitDataMap() procedure...
'**************************************
'**************************************

Public Const g_sFRED_INPUT_ID As String = "DataMap_1"
Public Const g_sREVWR_STATUS_INP_ID As String = "DataMap_2"
Public Const g_sDOC_TYPE_ADMIN As String = "Administrative Matters"
Public Const g_sCTX_07_ADDIN As String = _
    "CorpTax.Office.Excel.AddIn.Version2007"
Public Const g_sEMPTY_CTX_PLUG As String = "DC"
Public Const g_sSTATUS_DEFAULT As String = "Not Started"
Public Const g_sRVW_STATUS_APPVD As String = "Approved"
Public Const g_sTITLE_SUFFIX As String = "FRED"
Public Const g_sXL_MACRO_FILE_EXT As String = ".xlsm"

Public Const g_sSELECT_STD As String = "Select..."
Public Const g_sSELECT_SI_CODE As String = _
    "-- Please select SI code and Name --"
Public Const g_sSELECT_DOC_TYPE As String = _
    "-- Please select Document Type --"



Public Const g_sSP_PPTY_JURISDIC As String = "Jurisdiction"
Public Const g_sSP_PPTY_DOC_TYPE As String = "Document Type"
Public Const g_sSP_PPTY_TAX_YR As String = "Tax Year"
Public Const g_sSP_PPTY_ENT_SNM As String = "Entity and Short Name"
Public Const g_sSP_PPTY_SICODE_SINM As String = "SI Code and Name"
Public Const g_sSP_PPTY_EOP As String = "Period End Date"
Public Const g_sSP_PPTY_PREP_NM As String = "Fred Preparer Name"
Public Const g_sSp_PPTY_STATUS_PREP As String = "Fred Preparer Status"
Public Const g_sSP_PPTY_REVIEW_NM As String = "Fred Reviewer Name"
Public Const g_sSP_PPTY_STATUS_REVIEW As String = "Fred Reviewer Status"
Public Const g_sFMS_DIR As String = "Forms/"

Public Const g_ySP_METADATA_FLDS As Byte = 10
Public Const g_yBEG_POS_SICODE_IN_CTXCODE As Byte = 3
Public Const g_yLEN_SICODE As Byte = 4
Public Const g_yMAX_RECONNECT_ATTEMPTS As Byte = 3
Public Const g_dRVW_STATUS_ID_APPRVD As Double = 4
```

Globals - 2

```
Public g_comaddinCTX As COMAddIn

Public g_objCTXOffice As Object

Public g_cMBxHndlr As MsgBoxHandler

Public g_enExecMode As ProjExecutionMode

Public g_ffOrigFileSaveMode As XlFileFormat

Public g_sAcctNm As String

Public g_bExecuteWbkOnCloseCode As Boolean
Public g_bDebugMode As Boolean
Public g_bWbCheckedOutSP As Boolean
Public g_bInvalidEntCode As Boolean
Public g_bEntCodeNotOnSP As Boolean
Public g_bIAmReviewer As Boolean
Public g_bIAmSuperReviewer As Boolean
Public g_bSkipTrapForPreppingSomeoneElsesWork As Boolean
Public g_bReviewerRejectedFile As Boolean
Public g_bReviewerApproved As Boolean
Public g_bDevShutDownMode As Boolean


Public Enum ProjRole
    projPreparer
    projReviewer
End Enum

Public Enum ProjCTXRecalcObj
    projWsh
    projWkbk
End Enum

Public Enum ProjExecutionMode
    projProdActual
    projProdTest
    projDebugNoStops
    projDebugStepThru
End Enum
```

```
Main - 1

Option Explicit
'================================================================================
Private Sub PopulateArraySuperRvwrs(ByRef ArraySuprRvwrsRESULT() As String)
Const sCOL_TITLE As String = "Login Account"
Dim tblSuperRvwrs As ListObject
Dim yTblCols As Byte
Dim ySuperRvwrs As Byte
Dim yLogInNmCol As Byte
Dim i As Byte

'Define and refresh ListObject (= data link to SharePoint List)...
Set tblSuperRvwrs = WshSprRvwrsLnk.ListObjects(1)

'In case there are probs with the SP connection...
On Error Resume Next
tblSuperRvwrs.Refresh
On Error GoTo 0 '...restore normal error trapping

'Find which column contains the LogIn Name field...
yTblCols = tblSuperRvwrs.HeaderRowRange.Columns.Count
For i = 1 To yTblCols
    If StrComp(UCase(sCOL_TITLE), _
        UCase(tblSuperRvwrs.HeaderRowRange.Cells(1, i))) = 0 Then
        yLogInNmCol = i
        i = yTblCols '...break loop
    End If
Next i

'Size and populate the array...
ySuperRvwrs = tblSuperRvwrs.DataBodyRange.Rows.Count
ReDim ArraySuprRvwrsRESULT(0 To ySuperRvwrs - 1)
For i = 1 To ySuperRvwrs
    ArraySuprRvwrsRESULT(i - 1) = _
        tblSuperRvwrs.DataBodyRange.Cells(i, yLogInNmCol)
    Debug.Print "Rvwr"; i; ":"; ArraySuprRvwrsRESULT(i - 1)
Next i

End Sub
'================================================================================
Public Sub ProtectPreparerAndReviewerWshs()
Const yWSHS_TO_PROTECT As Byte = 2
Dim arrWshsToProtect(0 To yWSHS_TO_PROTECT - 1) As Worksheet
Dim i As Integer

Set arrWshsToProtect(0) = WshPreparer
Set arrWshsToProtect(1) = WshReviewer

For i = 1 To yWSHS_TO_PROTECT
    arrWshsToProtect(i - 1).Protect _
        Password:=g_sWSH_PWD, _
        userinterfaceonly:=True, _
        AllowFormattingColumns:=True, _
        AllowFormattingRows:=True, _
        AllowSorting:=True, _
        AllowFiltering:=True
Next i

End Sub
'================================================================================
Public Sub SetVisibilityOnAllButInstrucWshs()
Const yWSHS_TO_HIDE As Byte = 5
Dim arrwsToHide(0 To yWSHS_TO_HIDE - 1) As Worksheet
Dim ws As Worksheet
Dim enWshVisibility As XlSheetVisibility
Dim i As Integer

'Populate array...
Set arrwsToHide(0) = WshDataCache
Set arrwsToHide(1) = WshDropDown
Set arrwsToHide(2) = WshPreparer
Set arrwsToHide(3) = WshReviewer
```

```
Main - 2

Set arrwsToHide(4) = WshSprRvwrsLnk

'This SHOULD be unneeded, but just in case...
WshInstruction.Visible = xlSheetVisible
WshInstruction.Select

If g_enExecMode = projProdActual Then Application.ScreenUpdating = False

'Select sheet visibility based on execution mode...
Select Case g_enExecMode
    Case projProdActual
        enWshVisibility = xlSheetVeryHidden
    Case projProdTest
        enWshVisibility = xlSheetHidden
    Case projDebugNoStops, projDebugStepThru
        enWshVisibility = xlSheetVisible
End Select

'Apply the visibility to the sheets...
For i = 1 To yWSHS_TO_HIDE
    arrwsToHide(i - 1).Visible = enWshVisibility
Next i

ExitPoint:
    Application.ScreenUpdating = True
    Exit Sub
End Sub
'================================================================================
Public Sub VerifyCTXAndXL07AndPrepInstrWsh()
Const sXL_PRE_07 As String = "Pre-Office 2007"
Const sXL07 As String = "Office 2007"
Const sXL_POST_07 As String = "Office 2007+"
Const iXL07_VER As Integer = 12

Dim sXLVer As String

WshInstruction.Unprotect g_sWSH_PWD

'If here obviously our macros are working...
WshInstruction.Range("ptrMacrosEnabled") = "Yes"

'Check on XL/Office version...
Select Case Int(Val(Application.Version))
    Case Is < iXL07_VER
        sXLVer = sXL_PRE_07
    Case iXL07_VER
        sXLVer = sXL07
    Case Is > iXL07_VER
        sXLVer = sXL_POST_07
End Select

'Populate Instr wsh cell...
WshInstruction.Range("ptrXLVersion") = sXLVer

'Trap for user using a pre-2007 version of Office...
If Int(Val(Application.Version)) < 12 Then
    g_cMBxHndlr.SetIncorrectVersionMSOfficeMsg
    g_cMBxHndlr.ShowMsgBoxNoAnswer vbExclamation
    ThisWorkbook.Close SaveChanges:=False
    g_cMBxHndlr.SetRunningOnPreXL07Msg
    g_cMBxHndlr.ShowMsgBoxNoAnswer vbCritical
    Exit Sub
End If

'Fire up CTX add-in objs, trapping for the add-in not being installed...
On Error GoTo ErrorHandler:
If Application.COMAddIns(g_sCTX_07_ADDIN).Connect Then
    Set g_comaddinCTX = Application.COMAddIns.Item(g_sCTX_07_ADDIN)
    Set g_objCTXOffice = g_comaddinCTX.Object
End If
On Error GoTo 0 '...reset
```

```
'Now populate cells and enable button...
WshInstruction.Range("ptrCTXInstalled") = "Yes"
WshInstruction.Protect Password:=g_sWSH_PWD, userinterfaceonly:=True
Range("CTX_Enterprise").Formula = "=CtxDataProfileEnterprise()"
WshInstruction.btnUnhideAndPrepRoleBasedWsh.Enabled = True

'If here then...
g_bExecuteWbkOnCloseCode = True

ExitPoint:
    Exit Sub

ErrorHandler:
    g_cMBxHndlr.SetCTXAddInNotInstalledMsg
    g_cMBxHndlr.ShowMsgBoxNoAnswer vbExclamation
    ThisWorkbook.Close False
    GoTo ExitPoint '...should be superfluous
End Sub
'================================================================================
Public Function IsMemberFREDReviewerADGrp() As Boolean
Const sFRED_RVWRS_GRP_INIT_1 As String = _
    "LDAP://cn=fred_reviewers,ou=general, ou=groups, ou=newyork_ny, "
Const sFRED_RVWRS_GRP_INIT_2 As String = _
    "ou=corporate_systems,ou=r1,dc=r1-core,dc=r1,dc=aig,dc=net"

Dim objFREDReviewersGrp As Object
Dim varMember As Variant
Dim objUser As Object
Dim bIsMember As Boolean '...default = FALSE

If g_enExecMode = projDebugStepThru Then Stop
Set objFREDReviewersGrp = _
    GetObject(sFRED_RVWRS_GRP_INIT_1 & sFRED_RVWRS_GRP_INIT_2)

'**********
'*** Don't know what this does...
objFREDReviewersGrp.GetInfo
'**********

For Each varMember In objFREDReviewersGrp.Members
    Set objUser = GetObject(varMember.adspath)
    g_sAcctNm = objUser.samaccountname
    If objUser.Class <> "user" Then SID_to_user (objUser.cn)
    If StrComp(Environ("username"), g_sAcctNm, vbTextCompare) = 0 Then
        bIsMember = True
        GoTo ExitPoint
    End If
Next varMember

ExitPoint:
    'If in developer mode allow override of user-as-reviewer...
    If g_bDebugMode And Not bIsMember Then
        Dim ansBecomeRevwr As VbMsgBoxResult
        g_cMBxHndlr.vOverrideNotAReviewerQues
        ansBecomeRevwr = g_cMBxHndlr.ShowMsgBoxReturnAnswer(vbYesNo)
        If ansBecomeRevwr = vbYes Then bIsMember = True
    End If
    IsMemberFREDReviewerADGrp = bIsMember
    Exit Function
End Function
'================================================================================
Private Function IsUserMemberSuperReviewersGrp() As Boolean
'NOTE:  Chr(92) = backslash ("\")
Dim arrsSuperRvwrs() As String
Dim sUserLogin As String
Dim bUserIsSuperRvwr As Boolean
Dim i As Byte

sUserLogin = Environ("UserDomain") & Chr(92) & Environ("UserName")
```

```
Main - 4

PopulateArraySuperRvwrs ArraySuprRvwrsRESULT:=arrsSuperRvwrs
For i = 0 To UBound(arrsSuperRvwrs)
    'vbTextCompare => Case INsensitive...
    If StrComp(sUserLogin, arrsSuperRvwrs(i), vbTextCompare) = 0 Then
        bUserIsSuperRvwr = True
        i = UBound(arrsSuperRvwrs) '...end loop
    End If
Next i

IsUserMemberSuperReviewersGrp = bUserIsSuperRvwr

End Function
'================================================================================
Private Sub SID_to_user(SID As String)
Const sWMI_SVC_INIT_1 As String = "winmgmts:\\"
Const sWMI_SVC_INIT_2 As String = "\root\cimv2"
Const sCOMPUTER As String = "."
Const sWMI_SID_SVC_INIT As String = "Win32_SID.SID='"

Dim objWMIservice As Object
Dim objAccount As Object

Set objWMIservice = GetObject(sWMI_SVC_INIT_1 & sCOMPUTER & sWMI_SVC_INIT_2)

'NOTE:  Chr(39) = apostrophe (i.e., " ' ")...
Set objAccount = objWMIservice.get(sWMI_SID_SVC_INIT & SID & Chr(39))

g_sAcctNm = objAccount.AccountName

ExitPoint:
    Exit Sub

End Sub
'================================================================================
Public Sub DisplayRoleBasedWsh(Role As ProjRole)
Dim wsToShow As Worksheet
'In case user has displayed one wsh and has now changed his role...
Dim wsToHide As Worksheet
Dim visToApply As XlSheetVisibility

On Error GoTo ExitPoint
If g_enExecMode = projProdActual Or g_enExecMode = projProdTest Then _
    Application.ScreenUpdating = False

If Role = projReviewer Then
    Set wsToShow = WshReviewer
    Set wsToHide = WshPreparer
Else
    Set wsToShow = WshPreparer
    Set wsToHide = WshReviewer
End If
wsToShow.Visible = xlSheetVisible

'Set visibility of wshToHide...
Select Case g_enExecMode
    Case ProjExecutionMode.projProdActual
        visToApply = xlSheetVeryHidden
    Case ProjExecutionMode.projProdTest
        visToApply = xlSheetHidden
    Case Else
        visToApply = xlSheetVisible
End Select
wsToHide.Visible = visToApply

wsToShow.Select
'Deactivate, to be safe...
If Role = projReviewer Then WshReviewer.btnSubmitRvwrStatus.Enabled = False

wsToShow.Range("ptrCase").Select '...locally-scoped XL name

ExitPoint:
```

```
    Application.ScreenUpdating = True
    Exit Sub
End Sub
'================================================================================
Public Sub InitializeReviewerWsh()
Dim sUserNm As String
'Default for booleans = FALSE...
Dim bEnableSubmitStatusBtn As Boolean
Dim bNoReviewerPreviouslyAssigned As Boolean '...used for our error-trapping
Dim bUnLockInputCells As Boolean
Dim bUserIsSuperRvwr As Boolean '...used for debugging
Dim bRvwrApprovedWbk As Boolean

If g_enExecMode = projDebugStepThru Then Stop
InvokeCTXCalc projWkbk
If g_enExecMode = projDebugStepThru Then Stop

sUserNm = GetUserFullName()

'One last check, NOW testing that we have the correct CTX code in place _
[NOTE:  because we had some issues with our code here when a user was _
not logged in, we use this little bit of error handling]...
On Error Resume Next
g_bReviewerApproved = False '...just to reset matters
If (Range("ptrFREDRvwrStatusID").Value = g_dRVW_STATUS_ID_APPRVD) And _
        (Range("CTX_Status_Review").Value = g_dRVW_STATUS_ID_APPRVD) Then _
        bRvwrApprovedWbk = True
If bRvwrApprovedWbk Then
    g_bReviewerApproved = True
    'If here we also need to set this boolean...
    bUserIsSuperRvwr = IsUserMemberSuperReviewersGrp()
    'An override for development...
    If g_bIAmSuperReviewer Then bUserIsSuperRvwr = True
End If
On Error GoTo 0 '...restore matters

Main.SetWbkApprovedMsgVisibilityOnInstrucWsh MsgVisible:=g_bReviewerApproved

'Trap for an already-approved Wb (and reviewer is NOT a super-reviewer)...
If g_bReviewerApproved And Not bUserIsSuperRvwr Then
    g_cMBxHndlr.SetWbkAlreadyApprovedMsg
    g_cMBxHndlr.ShowMsgBoxNoAnswer vbInformation
    GoTo ExitPoint
End If

''Trap for a user who hasn't checked his file out of SP...
'If Not g_bWbCheckedOutSP Then
'    SetLockedStatusAndColorForXLRng _
'        XLRangeName:="Rev_StatusPlusVisOrgCells", _
'        LockRange:=True
'    GoTo ExitPoint
'End If

With WshReviewer
    'Trap for preparer as reviewer...
    If sUserNm = Range("Preparer_nm") Then
        g_cMBxHndlr.SetPreparerCannotReviewMsg
        g_cMBxHndlr.ShowMsgBoxNoAnswer vbInformation
        GoTo ExitPoint
    End If

    'Handle no reviewer yet assigned...
    If (IsEmpty(Range("Reviewer_nm"))) Then
        bNoReviewerPreviouslyAssigned = True
        Dim ansAreYouReviewer As VbMsgBoxResult
        g_cMBxHndlr.SetWillYouBeReviewerQues
        ansAreYouReviewer = g_cMBxHndlr.ShowMsgBoxReturnAnswer(vbYesNo)
        If ansAreYouReviewer = vbNo Then
            g_cMBxHndlr.SetReadOnlyAccessMsg
            g_cMBxHndlr.ShowMsgBoxNoAnswer vbInformation
            GoTo ExitPoint
```

```
        End If
        'If here user said YES to becoming reviewer...
        .Range("Reviewer_nm") = sUserNm
        'We are getting an unspecified error thrown if the user answers _
        "NO" to the "Overwrite Existing File" prompt.  Ergo, we need a little _
        error trap...
SavingWbkFirstTimeRvwr:
        On Error GoTo ErrorTrap
        ThisWorkbook.Save
        On Error GoTo 0 '...reset
        bUnLockInputCells = True
        bEnableSubmitStatusBtn = True
        GoTo ExitPoint
    End If

    'Trap for new/different reviewer...
    If StrComp(Range("Reviewer_nm"), sUserNm) <> 0 Then
        Dim ansChngNm As VbMsgBoxResult
        g_cMBxHndlr.SetChangeReviewersNameQues
        ansChngNm = g_cMBxHndlr.ShowMsgBoxReturnAnswer(vbYesNo)
        If ansChngNm = vbYes Then
            .Range("Reviewer_nm") = sUserNm
SavingWbkChngRvwr:
            'We are getting an unspecified error thrown if the user answers _
            "NO" to the "Overwrite Existing File" prompt.  Ergo...
            On Error GoTo ErrorTrap
            ThisWorkbook.Save
            On Error GoTo 0 '...reset
            g_cMBxHndlr.SetReviewerNameChangedMsg
            g_cMBxHndlr.ShowMsgBoxNoAnswer vbInformation
            bUnLockInputCells = True
            bEnableSubmitStatusBtn = True
        Else '...user does not want to become reviewer
            g_cMBxHndlr.SetReviewerNameNotChangedMsg
            g_cMBxHndlr.ShowMsgBoxNoAnswer vbInformation
        End If
        GoTo ExitPoint
    End If
End With

'If here then this is the simple case of a reviewer opening his own _
to-review worksheet...
bEnableSubmitStatusBtn = True
bUnLockInputCells = True

ExitPoint:
    SetLockedStatusAndColorForXLRng _
        XLRangeName:="Rev_StatusPlusVisOrgCells", _
        LockRange:=(Not bUnLockInputCells)
    WshReviewer.btnSubmitRvwrStatus.Enabled = bEnableSubmitStatusBtn
    Range("Status_Review").Select
    Exit Sub

ErrorTrap:
    Dim ansTrySaveFileAgain As VbMsgBoxResult
    g_cMBxHndlr.SetYouHaveNotSavedWbkQues
    ansTrySaveFileAgain = g_cMBxHndlr.ShowMsgBoxReturnAnswer(vbYesNo)
    If ansTrySaveFileAgain = vbYes Then
        If bNoReviewerPreviouslyAssigned Then
            GoTo SavingWbkFirstTimeRvwr
        Else
            GoTo SavingWbkChngRvwr
        End If
    Else '...vbNo
        GoTo ExitPoint
    End If

End Sub
'================================================================================
Public Sub InvokeCTXCalc(CalcObj As ProjCTXRecalcObj)
Const sPLS_WAIT As String = _
```

```
Main - 7

    "Please wait while CorpTax refreshes the data in this workbook"
Application.StatusBar = sPLS_WAIT

'Test for existence/validity of object...
If g_objCTXOffice Is Nothing Then
    Dim bAttemptSuccessful As Boolean
    Main.RecreateCTXOffObj RecreatedRESULT:=bAttemptSuccessful
    If Not bAttemptSuccessful Then GoTo InvalidCTXObj
End If

If CalcObj = projWsh Then
    g_objCTXOffice.CalculateWorksheet
Else '...wkbk
    g_objCTXOffice.CalculateWorkbook
End If

ExitPoint:
    Application.StatusBar = vbNullString
    Exit Sub

InvalidCTXObj:
    g_cMBxHndlr.SetCannotConnectToCTXOfficeMsg
    g_cMBxHndlr.ShowMsgBoxNoAnswer vbExclamation
    GoTo ExitPoint
End Sub
'================================================================================
Private Function GetUserFullName() As String
Const sSUNAM_LA_DOMAIN As String = "Sunamerica-LA"
Const sINVEST_DOMAIN As String = "Investments"
Const sSUNAM_CONN_STR_1 As String = "LDAP://CN="
Const sSUNAM_CONN_STR_2 As String = " ,cn=users,dc=sarm,dc=aigrs,dc=net"
Const sINVEST_CONN_STR_2 As String = _
    " ,ou=RegularUsers,ou=Employees,ou=Users,ou=US,ou=investments,"
Const sINVEST_CONN_STR_3 As String = "dc=investments,dc=aig,dc=net"

Dim objUser As Object
Dim sUserFullNm As String '...default = nullstring
Dim bInWINMgt As Boolean

'Added 7/16/2012...
Dim objSysInfo As Object

On Error GoTo OrigCode

Set objSysInfo = CreateObject("ADSystemInfo")
Set objUser = GetObject("LDAP://" & objSysInfo.UserName)
sUserFullNm = objUser.DisplayName
If StrComp(Trim(sUserFullNm), vbNullString) <> 0 Then GoTo ExitPoint

'*****
'From here through the remainder of the function is the code that was _
working prior to our 7/16/2012 modification (slightly updated)...
'*****
OrigCode:
'If here we may have thrown an error, so reset things...
Err.Clear

'Just in case we did manage to create the object...
If Not objUser Is Nothing Then Set objUser = Nothing

Dim objWINMgmt As Object
Dim varItem As Variant

On Error GoTo ErrorHandler

'Handle SunAmerica domain...
If StrComp(Environ("UserDomain"), sSUNAM_LA_DOMAIN, vbTextCompare) = 0 Then
    Set objUser = _
        GetObject(sSUNAM_CONN_STR_1 & Environ("username") & sSUNAM_CONN_STR_2)
    objUser.GetInfo
    sUserFullNm = objUser.DisplayName
```

```
    GoTo ExitPoint
End If

'Handle Investments domain...
If StrComp(Environ("UserDomain"), sINVEST_DOMAIN, vbTextCompare) = 0 Then
    Set objUser = _
        GetObject(sSUNAM_CONN_STR_1 & Environ("username") & _
            sINVEST_CONN_STR_2 & sINVEST_CONN_STR_3)
    objUser.GetInfo
    sUserFullNm = objUser.DisplayName
    GoTo ExitPoint
End If

'Handle "regular" domains...
bInWINMgt = True
Set objWINMgmt = _
    GetObject("WinMgmts:").instancesOf("Win32_NetworkLoginProfile")
For Each varItem In objWINMgmt
    If StrComp(Trim(varItem.FullName), vbNullString, vbTextCompare) <> 0 Then
        sUserFullNm = varItem.FullName
        GoTo ExitPoint
    End If
Next varItem

ExitPoint:
    If StrComp(Trim(sUserFullNm), vbNullString) = 0 Then GoTo ErrorHandler
    GetUserFullName = Trim(sUserFullNm)
    'NOTE:  We originally tried to destroy the objects created in this _
    function (objUser, objWINMgt, etc.) That code was here in ExitPoint. _
    However, none of the attempted methods - Close, _
    Dispose, End, Exit - was recognized.  We have therefore removed that _
    code.  If the objects are not destroyed upon exiting this funtion _
    presumably they will be destroyed when the wbk is closed...
    Exit Function

ErrorHandler:
    If bInWINMgt Then
        g_cMBxHndlr.SetWMINotInstalledMsg
    Else
        g_cMBxHndlr.SetCannotGetUserNameMsg
    End If
    g_cMBxHndlr.ShowMsgBoxNoAnswer vbCritical
    g_bExecuteWbkOnCloseCode = False '...since we won't be saving the Wbk
    ThisWorkbook.Close SaveChanges:=False

End Function
'================================================================================
Public Sub SetLockedStatusAndColorForXLRng(XLRangeName As String, _
    LockRange As Boolean)
Dim varColor As Variant

If LockRange Then
    varColor = 2 '...white
Else
    varColor = 6 '...yellow
End If

Range(XLRangeName).Select

With Selection
    .Interior.ColorIndex = varColor
    .Locked = LockRange
End With
End Sub
'================================================================================
Public Sub InitializePreparerWsh()
Dim sUserNm As String
Dim ansResetPreparer As VbMsgBoxResult
Dim bEnablePrepWshEditing As Boolean
Dim bAnotherPreparersWbk As Boolean
```

```
If g_enExecMode = projDebugStepThru Then Stop

SetLockedStatusAndColorForXLRng XLRangeName:="Input_Prep", LockRange:=True

'Leave the Ws locked up if the Reviewer status is Approved...
If g_bReviewerApproved Then GoTo ExitPoint

'If g_bWbCheckedOutSP Then
'    'Set a default for this procedure...
'    bEnablePrepWshEditing = True
'Else
'    'GoTo ExitPoint '...and leave bEnablePrepWshEditing = FALSE
'End If

bEnablePrepWshEditing = True

sUserNm = GetUserFullName()

'LOGIC: If user = preparer nothing will happen until ExitPoint...

'If no preparer has been defined declare the user as the preparer...
If IsEmpty(Range("Prep_nm")) Then
    Range("Prep_nm") = sUserNm
    GoTo ExitPoint
End If

'We have create this otherwise-unneeded boolean variable in order to _
handle matters during development...
If Range("Prep_nm") <> sUserNm Then bAnotherPreparersWbk = True
'In-development override...
If g_bSkipTrapForPreppingSomeoneElsesWork Then bAnotherPreparersWbk = False

'If here then we are handling the case of user opening another _
preparer's workbook...
If bAnotherPreparersWbk Then
    g_cMBxHndlr.SetChangePreparerToYouQues
    ansResetPreparer = g_cMBxHndlr.ShowMsgBoxReturnAnswer(vbYesNo)
    If ansResetPreparer = vbYes Then
        Range("Prep_nm") = sUserNm
        g_cMBxHndlr.SetYouAreNewPreparerMsg
        g_cMBxHndlr.ShowMsgBoxNoAnswer vbInformation
    Else
        g_cMBxHndlr.SetPreparerNotChangedMsg
        g_cMBxHndlr.ShowMsgBoxNoAnswer vbInformation
        'This is the one case for disabling buttons...
        bEnablePrepWshEditing = False
    End If
End If

ExitPoint:
    'Turn Wsh buttons on/off...
    With WshPreparer
        .btnUnlockInputRng.Enabled = bEnablePrepWshEditing
        .btnSubmitData.Enabled = bEnablePrepWshEditing
    End With
    Range("Input_Start").Select
    Exit Sub
End Sub
'================================================================================
Public Function IsFileInReview() As Boolean
Dim sRvwrStatus As String
IsFileInReview = False
sRvwrStatus = Range("Reviewer_Status")
If sRvwrStatus = "In Review" Or sRvwrStatus = "Approved" Then IsFileInReview = True
End Function
'================================================================================
Public Sub RecreateCTXOffObj(ByRef RecreatedRESULT As Boolean)

Dim i As Integer
RecreatedRESULT = False '...assuming the worst
```

```
Main - 10

If g_comaddinCTX Is Nothing Then _
    Set g_comaddinCTX = Application.COMAddIns.Item(g_sCTX_07_ADDIN)

For i = 1 To g_yMAX_RECONNECT_ATTEMPTS
    Set g_objCTXOffice = g_comaddinCTX.Object
    If Not g_objCTXOffice Is Nothing Then '... i.e., successful
        RecreatedRESULT = True
        i = g_yMAX_RECONNECT_ATTEMPTS
    End If
Next i
End Sub
'================================================================================
Public Sub EnsureValidPeriodEnd()
Dim iTaxYr As Integer

iTaxYr = CInt(Range("Period"))

'Trap for an entry which isn't even a date...
If Not IsDate(Range("Period_End")) Then
    g_cMBxHndlr.SetInvalidDateMsg
    g_cMBxHndlr.ShowMsgBoxNoAnswer vbInformation
    Range("Period_End") = DateSerial(iTaxYr, 12, 31)
    GoTo ExitPoint
End If

'If Period-End date occurs within the tax year we are good...
If Year(Range("Period_End")) = iTaxYr Then GoTo ExitPoint

'If here Period-End date occurred outside tax year.  Fix and notify user...
Dim iEOPYr As Integer
Dim iEOPMo As Integer
Dim iEOPDay As Integer

iEOPYr = Year(Range("Period_End"))
iEOPMo = Month(Range("Period_End"))
iEOPDay = Day(Range("Period_End"))

g_cMBxHndlr.SetChangingToValidYearMsg _
    YearEntered:=iEOPYr, _
    TaxYear:=iTaxYr
g_cMBxHndlr.ShowMsgBoxNoAnswer vbInformation

'Change to correct tax year year
Range("Period_End") = DateSerial(iTaxYr, iEOPMo, iEOPDay)

ExitPoint:
    Exit Sub

End Sub
'================================================================================
Public Sub SetWbkApprovedMsgVisibilityOnInstrucWsh(ByVal MsgVisible As Boolean)
Const varCOLOR_RED As Variant = 255 '...red

Dim wsOrig As Worksheet
Dim visWsOrig As XlSheetVisibility
Dim varFontColor As Variant
Dim bInstrucWshIsActive As Boolean

'Capture status...
Set wsOrig = ActiveSheet
visWsOrig = wsOrig.Visible

'Set font color that will be used...
If MsgVisible Then
    varFontColor = varCOLOR_RED
Else
    'Set font color = fill color - i.e., msg will be invisible...
    varFontColor = WshInstruction.Range("ptrWbkApproved").Interior.Color
End If

If StrComp(WshInstruction.Name, ActiveSheet.Name) = 0 Then
```

```
    bInstrucWshIsActive = True
Else
    Application.ScreenUpdating = False
    WshInstruction.Select
    WshInstruction.Visible = xlSheetVisible
End If

'Change the font color...
WshInstruction.Unprotect g_sWSH_PWD
WshInstruction.Range("ptrWbkApproved").Font.Color = varFontColor
WshInstruction.Protect _
    Password:=g_sWSH_PWD, _
    userinterfaceonly:=True, _
    AllowFormattingColumns:=True, _
    AllowFormattingRows:=True, _
    AllowSorting:=True, _
    AllowFiltering:=True

'Clean up if we were not on the Instruction ws to begin with...
If Not bInstrucWshIsActive Then
    'Restore matters...
    wsOrig.Select
    wsOrig.Visible = visWsOrig
    Application.ScreenUpdating = True
End If

End Sub
'================================================================================
Public Sub LockReviewerWshAfterReviewerApproval()
Dim visOrigRvwr As XlSheetVisibility
Dim sOrigActiveWshNm As String

Application.ScreenUpdating = False

'Capture status...
sOrigActiveWshNm = ActiveSheet.Name
visOrigRvwr = WshReviewer.Visible

WshReviewer.Visible = xlSheetVisible
WshReviewer.Select

ExitPoint:
    Application.ScreenUpdating = True
    Exit Sub
End Sub
'================================================================================
'================================================================================
```

```
'.....................................................................................
'.....................................................................................
'.......        Written by:             William H. White (consultant)
'.......        With:                   TEKSystems
'.......                                www.teksystems.com
'.......        For:                    AIG, Inc.
'.......                                Tax Technology
'.......                                1 WFC, 14th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         4/30/2012
'.......        Proj finished:          7/16/2012
'.....................................................................................
'.....................................................................................

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


'*************************************************************************************
'*************************************************************************************
'*************************  CLASS-LEVEL DECLARATIONS  ************************
'*************************************************************************************
'*************************************************************************************
Option Explicit
Option Compare Text  '...use IN-sensitive string comparisons

'**************************************
'MODULE/CLASS-WIDE USER-DEFINED TYPES
'**************************************
Private Type modSPPptyInfo
    modSPPpty As String
    modEmptyVal As String
    modSPCurrVal As String
    modXLCurrVal As String
    modCorrPptyVal As String
    modSPValIncorr As Boolean
End Type



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

'============================================================
'==================  (Private) FUNCTIONS  ==================
'============================================================
'=============================================================================
'=============================================================================
Private Function GetValidSPMetadataEntry(ByVal ShPtPrptyName As String, _
    ByVal EntryToTest As Variant, ByRef IsEntryValidRESULT) As Variant
Const sXML_ELEM_CLOSE As String = "/>"
Dim metaThisWkbkContTypPpts As MetaProperties
Dim varShPtFieldSchema As Variant
Dim lEntryPositionInSchema As Long
Dim lXMLElementClosePos As Long
Dim lXMLSegmentLength As Long
Dim bValidEntry As Boolean '...default = FALSE
Dim varMetaDataEntry As Variant

If EntryToTest = vbNull Then GoTo ExitPoint
```

```
'Set a default (between this assignment and default value for _
bValidEntry we are now assuming failure at the start)...
varMetaDataEntry = vbNullString

'Cast ContentTypeProperties as MetaProperties...
Set metaThisWkbkContTypPpts = ThisWorkbook.ContentTypeProperties

'SchemaXml is of type string.  However, we found that we were exceeding _
the size limitations of a string variable, so we have instead stored _
the SchemaXml as a variant...
varShPtFieldSchema = metaThisWkbkContTypPpts.SchemaXml
If varShPtFieldSchema = vbNullString Then GoTo ExitPoint

lEntryPositionInSchema = _
    InStr(1, varShPtFieldSchema, EntryToTest, vbTextCompare)
If lEntryPositionInSchema = 0 Then GoTo ExitPoint

lXMLElementClosePos = _
    InStr(lEntryPositionInSchema, varShPtFieldSchema, _
        sXML_ELEM_CLOSE, vbTextCompare)

lXMLSegmentLength = lXMLElementClosePos - lEntryPositionInSchema - 1
If lXMLSegmentLength < 1 Then GoTo ExitPoint     '...bad entry

'If here we apprear to be able to define a valid entry...
bValidEntry = True
varMetaDataEntry = _
    Mid(varShPtFieldSchema, lEntryPositionInSchema, lXMLSegmentLength)

ExitPoint:
    IsEntryValidRESULT = bValidEntry
    GetValidSPMetadataEntry = varMetaDataEntry
    Exit Function

End Function
'================================================================================
Private Function GetTrimmedCharsToLeftOfFirstHyphen( _
    ByVal HyphenatedString As String) As String
'Chr(45) = hyphen ("-")
Dim lHyphenLoc
lHyphenLoc = InStr(1, HyphenatedString, Chr(45))
'Trap for there being no hyphen...
If lHyphenLoc = 0 Then lHyphenLoc = Len(HyphenatedString) + 1
'NOTE:  if lHyphenLoc = 1 then this function will return an empty string...
GetTrimmedCharsToLeftOfFirstHyphen = _
    Trim(Left(HyphenatedString, lHyphenLoc - 1))
End Function
'================================================================================
Private Function GetTrimmedCharsToRightOfLastHyphen( _
    ByVal HyphenatedString As String) As String
'Chr(45) = hyphen ("-")
Dim lHyphenLoc
lHyphenLoc = InStrRev(HyphenatedString, Chr(45))
'Trap for there being no hyphen...
If lHyphenLoc = 0 Then lHyphenLoc = Len(HyphenatedString) + 1
'NOTE:  if lHyphenLoc = 1 then this function will return an empty string...
GetTrimmedCharsToRightOfLastHyphen = _
    Trim(Right(HyphenatedString, Len(HyphenatedString) - lHyphenLoc))
End Function



'============================================================
'==================  (Private) PROCEDURES  ==================
'============================================================
'================================================================================
Private Sub PopulateStrArrayWithSPPptyEnumeratedValues( _
    ByVal SPProperty As String, _
    ByRef ArrayEnumeratedShPtValsRESULT() As String)
Const sXML_ATTR_DISPLAYNAME As String = "ma:displayName="
Const sXML_ATTR_ENUMVAL As String = "enumeration value="
```

```
Const sXML_OPEN_TAG_RESTRIC As String = "<xsd:restriction"
Const sXML_CLOSE_TAG_RESTRIC As String = "</xsd:restriction>"
Const sXML_ELEM_CLOSE As String = "/>"
'NOTE:  Chr(34) = quotation mark (i.e., ")
Dim metaThisWkbkContTypPpts As MetaProperties
Dim varShPtFieldSchema As Variant
Dim sSearchFor As String
Dim sEnumVal As String
Dim lStartSearchPos As Long
Dim lDiscoveredTextPosInSchema As Long
Dim lOffsetFmXMLElemOpenChar As Long
Dim lEndPosEnum As Long
Dim lXMLElemCloseTagPos As Long
Dim lEnumValLen As Long
Dim iCurrArraySize As Integer
Dim i As Long

'Cast ContentTypeProperties as MetaProperties...
Set metaThisWkbkContTypPpts = ThisWorkbook.ContentTypeProperties

'Due to size of XML schema we capture it in a variant rather than in _
a string variable, which, we found, will not hold all of the data...
varShPtFieldSchema = metaThisWkbkContTypPpts.SchemaXml

'Since the XMLSchema can contain any number of enumerations, we first need _
to set the starting point of our search to the proper location (i.e., the _
<xsd:schema element> element which contains the enumeration we want...
lStartSearchPos = 1 '...our first search
sSearchFor = _
    sXML_ATTR_DISPLAYNAME & Chr(34) & SPProperty & Chr(34)
lDiscoveredTextPosInSchema = _
    InStr(lStartSearchPos, varShPtFieldSchema, sSearchFor)

'Reset where to begin searching...
lStartSearchPos = lDiscoveredTextPosInSchema
'Find where the <xsd:restriction> element begins...
sSearchFor = sXML_OPEN_TAG_RESTRIC
lDiscoveredTextPosInSchema = _
    InStr(lStartSearchPos, varShPtFieldSchema, sSearchFor)

'Reset where to begin searching...
lStartSearchPos = lDiscoveredTextPosInSchema
'Now find where the CLOSING (i.e., </xsd:restriction> element begins (we do _
this because we need to know when to stop searching for the enuneration _
values)...
sSearchFor = sXML_CLOSE_TAG_RESTRIC
lEndPosEnum = _
    InStr(lStartSearchPos, varShPtFieldSchema, sSearchFor)

'We didn't reset our start search position when looking for the closing _
XML tag.  Reset the start position now, so we get to the the first _
enumerated value...
sSearchFor = sXML_ATTR_ENUMVAL
lDiscoveredTextPosInSchema = _
    InStr(lStartSearchPos, varShPtFieldSchema, sSearchFor)

'Define our offset (add 1 to include quotation mark)...
lOffsetFmXMLElemOpenChar = Len(sXML_ATTR_ENUMVAL) + 1

'Clear out anything that might be in the array...
ReDim ArrayEnumeratedShPtValsRESULT(0 To 0) '...omitting PRESERVE!

'Populate the array...
Do
    'Move from 1st position in XML tag to 1st position of the attribute _
    VALUE (i.e., an enumerated value itself)...
    lDiscoveredTextPosInSchema = _
        lDiscoveredTextPosInSchema + lOffsetFmXMLElemOpenChar
    'Find position of closing tag (i.e., "/>")
    lStartSearchPos = lDiscoveredTextPosInSchema
    lXMLElemCloseTagPos = _
```

```
        InStr(lStartSearchPos, varShPtFieldSchema, sXML_ELEM_CLOSE)
    'Grab the value of the enumeration, then add it to the array...
    lEnumValLen = lXMLElemCloseTagPos - lStartSearchPos - 1
    sEnumVal = Mid(varShPtFieldSchema, lDiscoveredTextPosInSchema, lEnumValLen)
    ReDim Preserve ArrayEnumeratedShPtValsRESULT(0 To i)
    ArrayEnumeratedShPtValsRESULT(i) = sEnumVal
    i = i + 1
    'Move to next enumeration element...
    lDiscoveredTextPosInSchema = _
        InStr(lStartSearchPos, varShPtFieldSchema, sXML_ATTR_ENUMVAL)
    'Trap for being the last enumeration element in the schema...
    If lDiscoveredTextPosInSchema = 0 Then _
        lDiscoveredTextPosInSchema = lEndPosEnum
Loop While lDiscoveredTextPosInSchema < lEndPosEnum

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




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

'============================================================
'==================  (Public) FUNCTIONS  ====================
'============================================================
'================================================================================
Public Function IsXLEntryValidShPtEntry_Simple(ByVal XLWshEntry As String, _
    ByVal SPPptyName As String) As Boolean
Dim arrsShPtPptyEnumeratedVals() As String
Dim bValidEntry As Boolean '...FALSE
Dim i As Integer

PopulateStrArrayWithSPPptyEnumeratedValues _
    SPProperty:=SPPptyName, _
    ArrayEnumeratedShPtValsRESULT:=arrsShPtPptyEnumeratedVals

For i = 0 To UBound(arrsShPtPptyEnumeratedVals)
    If StrComp(XLWshEntry, arrsShPtPptyEnumeratedVals(i)) = 0 Then
        bValidEntry = True
        i = UBound(arrsShPtPptyEnumeratedVals) '...end FOR loop
    End If
Next i

IsXLEntryValidShPtEntry_Simple = bValidEntry

End Function
'================================================================================
Public Function IsXLEntryInSPEnumValsLeftMostChars(ByVal XLEntry As String, _
    ByVal SPPpty As String) As Boolean
Dim arrsSPEnumeratedVals() As String
Dim sSPEnumValLHS As String
Dim lXLEntry As Long
Dim i As Integer
Dim bXLEntryValid As Boolean '...FALSE

'Set number of chars we will search/trim...
lXLEntry = Len(XLEntry)

PopulateStrArrayWithSPPptyEnumeratedValues _
    SPProperty:=SPPpty, _
    ArrayEnumeratedShPtValsRESULT:=arrsSPEnumeratedVals

For i = 0 To UBound(arrsSPEnumeratedVals)
    'Get left-hand side of the SP enumeration...
    sSPEnumValLHS = Left(arrsSPEnumeratedVals(i), lXLEntry)
    'Compare...
```

```
    If StrComp(XLEntry, sSPEnumValLHS) = 0 Then
        bXLEntryValid = True
        i = UBound(arrsSPEnumeratedVals) '...end FOR loop
    End If
Next i

IsXLEntryInSPEnumValsLeftMostChars = bXLEntryValid

End Function
'================================================================================
Public Function GetFullSPPptyValueFromLHMostChars(ByVal LHMostChars As String, _
    ByVal SPPpty As String) As String
Dim arrsSPEnumeratedVals() As String
Dim sSPEnumValLHS As String
Dim sFullSPEnumValue As String
Dim lXLEntry As Long
Dim i As Integer

'Set number of chars we will search/trim...
lXLEntry = Len(LHMostChars)

PopulateStrArrayWithSPPptyEnumeratedValues _
    SPProperty:=SPPpty, _
    ArrayEnumeratedShPtValsRESULT:=arrsSPEnumeratedVals

For i = 0 To UBound(arrsSPEnumeratedVals)
    'Get left-hand side of the SP enumeration...
    sSPEnumValLHS = Left(arrsSPEnumeratedVals(i), lXLEntry)
    'Compare...
    If StrComp(LHMostChars, sSPEnumValLHS) = 0 Then
        sFullSPEnumValue = arrsSPEnumeratedVals(i)
        i = UBound(arrsSPEnumeratedVals) '...end FOR loop
    End If
Next i

GetFullSPPptyValueFromLHMostChars = sFullSPEnumValue

End Function
'================================================================================
Public Function GetFullSPPptyValueFromRHMostChars(ByVal RHMostChars As String, _
    ByVal SPPpty As String) As String
Dim arrsSPEnumeratedVals() As String
Dim sSPEnumValRHS As String
Dim sFullSPEnumValue As String
Dim lXLEntry As Long
Dim i As Integer

'Set number of chars we will search/trim...
lXLEntry = Len(RHMostChars)

PopulateStrArrayWithSPPptyEnumeratedValues _
    SPProperty:=SPPpty, _
    ArrayEnumeratedShPtValsRESULT:=arrsSPEnumeratedVals

For i = 0 To UBound(arrsSPEnumeratedVals)
    'Get right-hand side of the SP enumeration...
    sSPEnumValRHS = Right(arrsSPEnumeratedVals(i), lXLEntry)
    'Compare...
    If StrComp(RHMostChars, sSPEnumValRHS) = 0 Then
        sFullSPEnumValue = arrsSPEnumeratedVals(i)
        i = UBound(arrsSPEnumeratedVals) '...end FOR loop
    End If
Next i

GetFullSPPptyValueFromRHMostChars = sFullSPEnumValue

End Function
'================================================================================
```

```
'============================================================
'=================  (Public) PROCEDURES  ==================
'============================================================
'================================================================================
Public Sub NotifyUserOfIncorrectTaxYearSelection()

'Create a list of valid years we can pass into a MsgBox [currently SP _
property allows for an input of "Permanent", which we do NOT want to allow _
user to utilize - 5/15/12]...
Dim arrsValidYrs() As String
Dim sYrListForMsgBox As String
Dim i As Integer
PopulateStrArrayWithSPPptyEnumeratedValues _
    SPProperty:=g_sSP_PPTY_TAX_YR, _
    ArrayEnumeratedShPtValsRESULT:=arrsValidYrs
For i = 0 To UBound(arrsValidYrs)
    If IsNumeric(arrsValidYrs(i)) Then _
        sYrListForMsgBox = sYrListForMsgBox & vbTab & arrsValidYrs(i) & vbCr
Next i

g_cMBxHndlr.SetInvalidTaxYearMsg sYrListForMsgBox
g_cMBxHndlr.ShowMsgBoxNoAnswer vbInformation

ExitPoint:
    Exit Sub

End Sub
'================================================================================
Public Sub UpdateEndOfPeriodForNewTaxYear()
Dim iNewTaxYear As Integer
Dim iEOPMo As Integer
Dim iEOPDay As Integer

iNewTaxYear = CInt(Range("Period"))
iEOPMo = Month(Range("Period_End"))
iEOPDay = Day(Range("Period_End"))

g_cMBxHndlr.SetUpdatingEndOfPeriodMsg iNewTaxYear
g_cMBxHndlr.ShowMsgBoxNoAnswer vbInformation

'Change to correct tax year year
Range("Period_End") = DateSerial(iNewTaxYear, iEOPMo, iEOPDay)

End Sub
'================================================================================
Public Sub SetDocTitle()
Dim bTitleAlreadyExists As Boolean
Dim sCurrDocTtl As String
Dim sNewDocTtl As String
'NOTE:  chr(95) = underscore (i.e., "_")

'Skip if our "entity code" is the empty CTX code plug...
If StrComp(Range("CTX_Ent_Code"), g_sEMPTY_CTX_PLUG) = 0 Then GoTo ExitPoint

sCurrDocTtl = Trim(ThisWorkbook.BuiltinDocumentProperties("Title"))
sNewDocTtl = _
    Trim(Range("Period")) & Chr(95) & _
    Trim(Range("CTX_Ent_Code")) & Chr(95) & _
    g_sTITLE_SUFFIX

If StrComp(sCurrDocTtl, vbNullString) <> 0 Then bTitleAlreadyExists = True

'Tell user about setting/updating title...
g_cMBxHndlr.SetSettingDocumentTitleMsg _
    TitleAlreadyExists:=bTitleAlreadyExists, _
    NewTitle:=sNewDocTtl
If bTitleAlreadyExists Then
    'Give user the option to cancel our change...
    Dim ansProceed As VbMsgBoxResult
    ansProceed = g_cMBxHndlr.ShowMsgBoxReturnAnswer(vbOKCancel)
    If ansProceed = vbOK Then
```

```
        ThisWorkbook.BuiltinDocumentProperties("Title") = sNewDocTtl
    Else 'i.e., user canceled...
        g_cMBxHndlr.SetDocTitleNotChangedMsg sCurrDocTtl
        g_cMBxHndlr.ShowMsgBoxNoAnswer vbInformation
    End If
Else
    'Just tell user we are making the change, then do it...
    g_cMBxHndlr.ShowMsgBoxNoAnswer vbInformation
    ThisWorkbook.BuiltinDocumentProperties("Title") = sNewDocTtl
End If

ExitPoint:
    Exit Sub
End Sub
'================================================================================
Public Sub ClearEnumeratedShPtPpty(ByVal SPPpty As String)
Dim sEmptyEnumeratedSPPpty As String

'Set SP value to whatever prompt is appropriate (enumerated properties only)...
If StrComp(SPPpty, g_sSP_PPTY_SICODE_SINM) = 0 Then
    sEmptyEnumeratedSPPpty = g_sSELECT_SI_CODE
Else
    sEmptyEnumeratedSPPpty = g_sSELECT_STD
End If
'Set the property...
ThisWorkbook.ContentTypeProperties(SPPpty).Value = sEmptyEnumeratedSPPpty
End Sub
'================================================================================
Public Sub SynchShPtPropertiesWithXLEntries()
Const sTO As String = "to..."
Const sBLNK As String = "[Blank]"

'We have to handle the End-of-Period field seperately because the property _
returns a Date type, whereas all of the other SP fields return strings...
Dim arrtypSPPpts(0 To g_ySP_METADATA_FLDS - 2) As modSPPptyInfo
Dim sCorrectedSPPptyVal As String
Dim sSPPptyNm As String
Dim sSPPptsWeChanged As String
Dim bIsXLEntryValid As Boolean
Dim bSPPptyCorrect As Boolean
Dim bWeChangedSPPpty As Boolean
Dim i As Byte

'Populate array...
arrtypSPPpts(0).modSPPpty = g_sSP_PPTY_ENT_SNM
arrtypSPPpts(0).modEmptyVal = g_sSELECT_STD
arrtypSPPpts(0).modXLCurrVal = Range("CTX_Ent_Code")

arrtypSPPpts(1).modSPPpty = g_sSP_PPTY_TAX_YR
arrtypSPPpts(1).modEmptyVal = g_sSELECT_STD
arrtypSPPpts(1).modXLCurrVal = Range("Period")

arrtypSPPpts(2).modSPPpty = g_sSP_PPTY_JURISDIC
arrtypSPPpts(2).modEmptyVal = g_sSELECT_STD
arrtypSPPpts(2).modXLCurrVal = Range("Jurisdiction")

arrtypSPPpts(3).modSPPpty = g_sSP_PPTY_DOC_TYPE
arrtypSPPpts(3).modEmptyVal = g_sSELECT_DOC_TYPE
arrtypSPPpts(3).modXLCurrVal = Range("Doc_Type")

arrtypSPPpts(4).modSPPpty = g_sSP_PPTY_SICODE_SINM
arrtypSPPpts(4).modEmptyVal = g_sSELECT_SI_CODE
arrtypSPPpts(4).modXLCurrVal = Range("SI_w_Name")

arrtypSPPpts(5).modSPPpty = g_sSP_PPTY_PREP_NM
arrtypSPPpts(5).modEmptyVal = vbNullString
arrtypSPPpts(5).modXLCurrVal = Range("Prep_nm")

arrtypSPPpts(6).modSPPpty = g_sSp_PPTY_STATUS_PREP
arrtypSPPpts(6).modEmptyVal = vbNullString
arrtypSPPpts(6).modXLCurrVal = Range("Status_Prep")
```

```
arrtypSPPpts(7).modSPPpty = g_sSP_PPTY_REVIEW_NM
arrtypSPPpts(7).modEmptyVal = vbNullString
arrtypSPPpts(7).modXLCurrVal = Range("Reviewer_nm")

arrtypSPPpts(8).modSPPpty = g_sSP_PPTY_STATUS_REVIEW
arrtypSPPpts(8).modEmptyVal = vbNullString
arrtypSPPpts(8).modXLCurrVal = Range("Status_Review")

'Populate array with current SP Ppty values...
For i = 1 To g_ySP_METADATA_FLDS - 1 '...-1 since we are skipping the EOP ppty
    arrtypSPPpts(i - 1).modSPCurrVal = _
        ThisWorkbook.ContentTypeProperties(arrtypSPPpts(i - 1).modSPPpty).Value
Next i

'Adjust some of the values, since there is not a direct comparison _
between the SP Ppty value and the XL value...
For i = 1 To g_ySP_METADATA_FLDS - 1 '...-1 since we are skipping the EOP ppty
    Select Case arrtypSPPpts(i - 1).modSPPpty
        Case g_sSP_PPTY_ENT_SNM
            'Convert entity and short name to CTX entity only...
            arrtypSPPpts(i - 1).modSPCurrVal = _
                GetTrimmedCharsToLeftOfFirstHyphen( _
                    arrtypSPPpts(i - 1).modSPCurrVal)
        Case g_sSP_PPTY_SICODE_SINM
            'Convert SI Code and name in SP to SI Code, and then _
            CTX code in XL to SI code...
            arrtypSPPpts(i - 1).modSPCurrVal = _
                GetTrimmedCharsToLeftOfFirstHyphen( _
                    arrtypSPPpts(i - 1).modSPCurrVal)
            arrtypSPPpts(i - 1).modXLCurrVal = _
                Mid(Range("CTX_Ent_Code"), _
                    g_yBEG_POS_SICODE_IN_CTXCODE, g_yLEN_SICODE)
        Case g_sSP_PPTY_DOC_TYPE
            arrtypSPPpts(i - 1).modSPCurrVal = _
                GetTrimmedCharsToRightOfLastHyphen( _
                    arrtypSPPpts(i - 1).modSPCurrVal)
            'OVERRIDE WHAT'S IN EXCEL - JUST TEST THAT SP VAL = OUR CONSTANT...
            arrtypSPPpts(i - 1).modXLCurrVal = g_sDOC_TYPE_ADMIN
    End Select
Next i

'Loop through each SP property and see if it is carrying the correct value...
For i = 1 To g_ySP_METADATA_FLDS - 1 '...-1 since we are skipping the EOP ppty
    If StrComp(arrtypSPPpts(i - 1).modSPCurrVal, _
        arrtypSPPpts(i - 1).modXLCurrVal) <> 0 Then
        'Set correct SP Ppty value, first handling special cases...
        Select Case arrtypSPPpts(i - 1).modSPPpty
            Case g_sSP_PPTY_ENT_SNM, g_sSP_PPTY_SICODE_SINM
                arrtypSPPpts(i - 1).modCorrPptyVal = _
                    GetFullSPPptyValueFromLHMostChars( _
                        arrtypSPPpts(i - 1).modXLCurrVal, _
                        arrtypSPPpts(i - 1).modSPPpty)
            Case g_sSP_PPTY_DOC_TYPE
                arrtypSPPpts(i - 1).modCorrPptyVal = _
                    GetFullSPPptyValueFromRHMostChars( _
                        arrtypSPPpts(i - 1).modXLCurrVal, _
                        arrtypSPPpts(i - 1).modSPPpty)
            Case Else
                arrtypSPPpts(i - 1).modCorrPptyVal = _
                    arrtypSPPpts(i - 1).modXLCurrVal
        End Select
        'Set two flags...
        If Not bWeChangedSPPpty Then
            bWeChangedSPPpty = True
            Dim sNewSPPpty As String
        End If
        arrtypSPPpts(i - 1).modSPValIncorr = True
    End If
    'Now build the string we will pass into our message box...
    If arrtypSPPpts(i - 1).modSPValIncorr Then
```

```
        Select Case arrtypSPPpts(i - 1).modSPPpty
            Case g_sSP_PPTY_REVIEW_NM, g_sSP_PPTY_PREP_NM
                sNewSPPpty = sBLNK
            Case Else
                sNewSPPpty = arrtypSPPpts(i - 1).modCorrPptyVal
        End Select
        'Create/build msgbox prompt...
        sSPPptsWeChanged = _
            sSPPptsWeChanged & vbTab & arrtypSPPpts(i - 1).modSPPpty & _
            vbTab & sTO & vbTab & sNewSPPpty & vbCr
    End If
Next i

'Check EOP...
If ThisWorkbook.ContentTypeProperties(g_sSP_PPTY_EOP).Value <> _
        Range("Period_End") Then
    'Set flag...
    If Not bWeChangedSPPpty Then bWeChangedSPPpty = True
    'Add note to message box string...
    sSPPptsWeChanged = sSPPptsWeChanged & vbTab & g_sSP_PPTY_EOP & _
            vbTab & sTO & vbTab & _
            Format(Range("Period_End"), "mm/dd/yyyy") & vbCr
End If

'Check for an empty Document title...
If StrComp(ThisWorkbook.BuiltinDocumentProperties("Title"), _
        vbNullString) = 0 And _
        StrComp(g_sEMPTY_CTX_PLUG, Range("CTX_Ent_Code")) <> 0 Then
    'Set flag...
    If Not bWeChangedSPPpty Then bWeChangedSPPpty = True
    'Add note to message box string...
    Dim sNewDocTtl As String
    sNewDocTtl = _
        Trim(Range("Period")) & Chr(95) & _
        Trim(Range("CTX_Ent_Code")) & Chr(95) & _
        g_sTITLE_SUFFIX
    sSPPptsWeChanged = sSPPptsWeChanged & vbTab & "Title" & _
            vbTab & sTO & vbTab & sNewDocTtl
End If

If Not bWeChangedSPPpty Then GoTo ExitPoint

'If here we need to update some SP properties...
g_cMBxHndlr.SetCorrectingSPPptsMsg sSPPptsWeChanged
g_cMBxHndlr.ShowMsgBoxNoAnswer vbInformation
For i = 1 To g_ySP_METADATA_FLDS - 1 '...-1 since we are skipping the EOP ppty
    If arrtypSPPpts(i - 1).modSPValIncorr Then _
        ThisWorkbook.ContentTypeProperties( _
            arrtypSPPpts(i - 1).modSPPpty).Value = _
            arrtypSPPpts(i - 1).modCorrPptyVal
Next i

'Handle EOP...
If ThisWorkbook.ContentTypeProperties(g_sSP_PPTY_EOP).Value <> _
    Range("Period_End") Then _
        ThisWorkbook.ContentTypeProperties(g_sSP_PPTY_EOP).Value = _
            Range("Period_End").Value

ExitPoint:
    Exit Sub

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

```
vDevelopment - 1

Option Explicit
'================================================================================
'================================================================================
Private Sub vPrintXMLSchemaToFile()
'Use this procedure to generate the latest XML schema behind the SharePoint _
site where FRED resides.  It wouldn't hurt to run this procedure whenever _
the allowable entries in the SharePoint metadata fields are updated.  The _
file you create is NOT directly accessed by the VBA in this file.  However, _
if you need to take a look at what's available to choose from this file is _
a convenient source...
Const sPATH As String = "L:\TaxTech\Sharepoint\ctx\"
Const sXML_SCHM_FILE_NM As String = "FREDShPtXMLSchemaContent.xml"
Dim metaThisWkbkContTypPpts As MetaProperties
Dim varShPtFieldSchema As Variant
Dim iFile As Integer
Dim i As Integer

Set metaThisWkbkContTypPpts = ThisWorkbook.ContentTypeProperties
varShPtFieldSchema = metaThisWkbkContTypPpts.SchemaXml

iFile = FreeFile()
Open sPATH & sXML_SCHM_FILE_NM For Output As #iFile
Print #iFile, varShPtFieldSchema
Close #iFile
End Sub
'================================================================================
Private Sub vPrintShPtPpptsToImmediateWindow()
Dim yNmbrProps As Byte
Dim i As Byte

yNmbrProps = ThisWorkbook.ContentTypeProperties.Count
For i = 1 To yNmbrProps
    Debug.Print "Ppty " & i & ":  " & _
        ThisWorkbook.ContentTypeProperties(i).Name
Next i

End Sub
'================================================================================
Private Sub vPrintBuiltInDocPropsToImmediateWindow()
Dim yNmbrProps As Byte
Dim i As Byte

yNmbrProps = ThisWorkbook.BuiltinDocumentProperties.Count
For i = 1 To yNmbrProps
    Debug.Print "Ppty " & i & ":  " & _
        ThisWorkbook.BuiltinDocumentProperties(i).Name
Next i

End Sub
'================================================================================
Public Sub vUnhideAndUnprotectWshs()
Const yNO_WSHS As Byte = 6
Dim arrWshs(0 To yNO_WSHS - 1) As Worksheet
Dim i As Byte

Set arrWshs(0) = WshPreparer
Set arrWshs(1) = WshReviewer
Set arrWshs(2) = WshDropDown
Set arrWshs(3) = WshDataCache
Set arrWshs(4) = WshInstruction
Set arrWshs(5) = WshSprRvwrsLnk

For i = 1 To yNO_WSHS
    arrWshs(i - 1).Visible = xlSheetVisible
    arrWshs(i - 1).Unprotect g_sWSH_PWD
Next i
WshReviewer.Select
Range("colMatch_Rvwr").EntireColumn.Hidden = False
WshInstruction.Select
MsgBox "Done", vbOKOnly, "Perfectus"
End Sub
```

```
vDevelopment - 2

'================================================================================
Public Sub vPrepForShutDown()
Const sDEFAULT_ROLE As String = "Preparer"
Const sVER As String = "Version: "
Const yBKGRD_WSHS As Byte = 3
Const yUSER_WSHS As Byte = 3

Dim wsToPrep As Worksheet
Dim i As Byte
Dim ansUpdateVersionOnInstrWsh As VbMsgBoxResult

g_bDevShutDownMode = True

g_cMBxHndlr.vSetUpdateVersionDateOnDashboardQues
ansUpdateVersionOnInstrWsh = g_cMBxHndlr.ShowMsgBoxReturnAnswer(vbYesNo)

WshInstruction.Select

Dim arrwsBckgrd(0 To yBKGRD_WSHS) As Worksheet
Dim arrwsUser(0 To yUSER_WSHS) As Worksheet

Set arrwsBckgrd(0) = WshDataCache
Set arrwsBckgrd(1) = WshDropDown
Set arrwsBckgrd(2) = WshSprRvwrsLnk

'Keep the order!!!...
Set arrwsUser(0) = WshPreparer
Set arrwsUser(1) = WshReviewer
Set arrwsUser(2) = WshInstruction

For i = 1 To yBKGRD_WSHS
    arrwsBckgrd(i - 1).Visible = xlSheetVeryHidden
Next i

For i = 1 To yUSER_WSHS
    arrwsUser(i - 1).Visible = xlSheetVisible
    Select Case arrwsUser(i - 1).Name
        Case WshPreparer.Name
            Set wsToPrep = WshPreparer
            Dim iLastYr As Integer
            iLastYr = Year(Now()) - 1
            wsToPrep.Select
            Main.SetLockedStatusAndColorForXLRng _
                XLRangeName:="Input_Prep", _
                LockRange:=False
            Range("CTX_Ent_Code") = g_sEMPTY_CTX_PLUG
            Range("Period") = iLastYr
            Range("Period_End") = DateSerial(iLastYr, 12, 31)
            Range("Prep_nm").ClearContents
            Range("Status_Prep") = g_sSTATUS_DEFAULT '..."Not Started"
            WshPreparer.Range("ptrVisOrgChartVfd") = "No"
            Main.InvokeCTXCalc projWsh
            With WshPreparer
                .btnUnlockInputRng.Enabled = False
                .btnSubmitData.Enabled = False
            End With
            wsToPrep.Outline.ShowLevels 1, 1
            Main.SetLockedStatusAndColorForXLRng _
                XLRangeName:="Input_Prep", _
                LockRange:=True
            wsToPrep.Protect _
                Password:=g_sWSH_PWD, _
                userinterfaceonly:=True, _
                AllowFormattingColumns:=True, _
                AllowFormattingRows:=True, _
                AllowSorting:=True, _
                AllowFiltering:=True
            WshInstruction.Select
            wsToPrep.Visible = xlSheetVeryHidden
            Set wsToPrep = Nothing
        Case WshReviewer.Name
```

```
vDevelopment - 3

            Set wsToPrep = WshReviewer
            wsToPrep.Select
            wsToPrep.Unprotect g_sWSH_PWD
            WshReviewer.Range("ptrRvwrWsh_PrepStatus").Calculate
            SetLockedStatusAndColorForXLRng _
                XLRangeName:="Rev_StatusPlusVisOrgCells", _
                LockRange:=False
            Range("Status_Review") = g_sSTATUS_DEFAULT
            WshReviewer.Range("ptrVisOrgChartVfd") = "No"
            Range("Reviewer_nm") = vbNullString
            Main.InvokeCTXCalc projWsh
            WshReviewer.btnSubmitRvwrStatus.Enabled = False
            Range("colMatch_Rvwr").EntireColumn.Hidden = True
            wsToPrep.Outline.ShowLevels 1, 1
            SetLockedStatusAndColorForXLRng _
                XLRangeName:="Rev_StatusPlusVisOrgCells", _
                LockRange:=True
            wsToPrep.Protect _
                Password:=g_sWSH_PWD, _
                userinterfaceonly:=True, _
                AllowFormattingColumns:=True, _
                AllowFormattingRows:=True, _
                AllowSorting:=True, _
                AllowFiltering:=True
            WshInstruction.Select
            wsToPrep.Visible = xlSheetVeryHidden
            Set wsToPrep = Nothing
        Case WshInstruction.Name
            Set wsToPrep = WshInstruction
            Main.SetWbkApprovedMsgVisibilityOnInstrucWsh MsgVisible:=False
            wsToPrep.Select
            WshInstruction.Unprotect g_sWSH_PWD
            Range("Selected_Role") = sDEFAULT_ROLE
            If ansUpdateVersionOnInstrWsh = vbYes Then
                Dim yy As String
                Dim mm As String
                Dim dd As String
                yy = Format(Year(Now()), "0000")
                mm = Format(Month(Now()), "00")
                dd = Format(Day(Now()), "00")
                WshInstruction.Range("ptrVersionStamp") = _
                    sVER & yy & Chr(46) & mm & Chr(46) & dd
            End If
            wsToPrep.Protect _
                Password:=g_sWSH_PWD, _
                userinterfaceonly:=True, _
                AllowFormattingColumns:=True, _
                AllowFormattingRows:=True, _
                AllowSorting:=True, _
                AllowFiltering:=True
            Set wsToPrep = Nothing
    End Select
Next i

ThisWorkbook.BuiltinDocumentProperties("Title") = vbNullString

g_bDevShutDownMode = False '...reset
If g_cMBxHndlr Is Nothing Then Set g_cMBxHndlr = New MsgBoxHandler
g_cMBxHndlr.vSetPrepForShutDownCompletedMsg
g_cMBxHndlr.ShowMsgBoxNoAnswer vbInformation
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_cMBxHndlr Is Nothing Then
    Set g_cMBxHndlr = New MsgBoxHandler
End If
g_cMBxHndlr.vSetOmitPrintAreaTitlesMsg
ansOmitPrintStuff = g_cMBxHndlr.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 _
```

```
vDevelopment - 5

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

zTest - 1

```
Option Explicit
'================================================================================
'================================================================================
'================================================================================
'================================================================================
'================================================================================
```

MsgBoxHandler - 1

```
'.....................................................................................
'.....................................................................................
'.......        Written by:             William H. White (consultant)
'.......        With:                   TEKSystems
'.......                                www.teksystems.com
'.......        For:                    AIG, Inc.
'.......                                Tax Technology
'.......                                1 WFC, 14th floor
'.......        Current contact:        william.white@aig.com
'.......                                (212) 551-5846
'.......        Permanent contact:      www.rcpconsulting.biz
'.......                                billwhite@rcpconsulting.biz
'.......                                New Providence, NJ
'.......        Module created:         4/30/2012
'.......        Proj finished:          7/16/2012
'.....................................................................................
'.....................................................................................

'.....................................................................................
'.........................  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  ===================
'============================================================

'**************************************
'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 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)
```

```
MsgBoxHandler - 3


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
'================================================================================
Public Sub SetCTXAddInNotInstalledMsg()
m_sTitle = "CorpTax Add-In Not Found"
m_sPrompt = _
    "This application requires an installation of the CorpTax Office " & _
    "Excel add-in.  Please contact your local admin to have the add-in " & _
    "installed" & m_s3Crs & _
    "This workbook will now shut down."
End Sub
'================================================================================
Public Sub SetIncorrectVersionMSOfficeMsg()
m_sTitle = "Incorrect Version of MS Office"
m_sPrompt = _
    "You do not have correct version of Microsoft Office installed.  " & _
    "Please upgrade it to Office 2007 or later.  " & m_s2Crs & _
    "As an alternative you can also log onto Citrix and use the " & _
    "correct version of Office to run this file." & m_s3Crs & _
    "In the meantime this workbook will now shut down."
End Sub
'================================================================================
Public Sub SetNotLoggedIntoCTXMsg()
m_sTitle = "Attention Required"
m_sPrompt = _
    "The error message you just received occurred because you are not " & _
    "logged into CorpTax for Excel, or because you have not set up " & _
    "valid data profile." & m_s2Crs & _
    "Please handle these issue(s) before proceeding."
End Sub
'================================================================================
Public Sub SetWkbkNotChkdOutMsg()
m_sTitle = "Workbook Not Checked-Out of SharePoint"
m_sPrompt = _
    "This workbook has not been Checked Out on SharePoint." & m_s2Crs & _
    "Within the limitations that would otherwise apply you will be able " & _
    "to review this Excel file.  However, you will NOT be able to " & _
    "change this file nor to submit any of its data."
End Sub
'================================================================================
Public Sub SetNotAReviewerMsg()
m_sTitle = "Not a Reviewer"
m_sPrompt = _
    "'You are NOT included in the database as Reviewer.  Please contact " & _
    "the Tax Technology Team if this is an error.  "
End Sub
'================================================================================
Public Sub SetWMINotInstalledMsg()
m_sTitle = "Windows Management Instrumentation"
m_sPrompt = _
    "WMI has not been installed.  Please have your local desktop " & _
    "support troubleshoot the issue." & m_s2Crs & _
    "This workbook will now close."
```

```
MsgBoxHandler - 4

End Sub
'================================================================================
Public Sub SetPreparerCannotReviewMsg()
m_sTitle = "Review Prohibited"
m_sPrompt = _
    "Since you were the preparer you cannot be the reviewer." & m_s2Crs & _
    "You have read-only access."
End Sub
'================================================================================
Public Sub SetChangeReviewersNameQues()
m_sTitle = "Reset Reviewer's Name?"
m_sPrompt = "Do you want to change the Reviewer name as yours?"
End Sub
'================================================================================
Public Sub SetReviewerNameChangedMsg()
m_sTitle = "Name Changed"
m_sPrompt = "The Reviewer name is changed to yours."
End Sub
'================================================================================
Public Sub SetReviewerNameNotChangedMsg()
m_sTitle = "Name Unchanged"
m_sPrompt = "You choose not to update the Reviewer name." & m_s2Crs & _
    "You will have read-only access. "
End Sub
'================================================================================
Public Sub SetChangePreparerToYouQues()
m_sTitle = "Change Preparer to You?"
m_sPrompt = _
    "Someone else has already worked on this template. Do you want to " & _
    "override the preparer name with yours?"
End Sub
'================================================================================
Public Sub SetYouAreNewPreparerMsg()
m_sTitle = "Preparer Name Changed"
m_sPrompt = "You are now listed as the preparer."
End Sub
'================================================================================
Public Sub SetPreparerNotChangedMsg()
m_sTitle = "Preparer Not Changed"
m_sPrompt = "You chose not to update the preparer name." & m_s2Crs & _
    "You have read-only access."
End Sub
'================================================================================
Public Sub SetAlreadyInReviewMsg()
m_sTitle = "Currently in Review"
'NOTE:  chr(34) = quotation mark ("); chr(46) = period (.)
m_sPrompt = _
    "You cannot unlock input cells once the workbook is " & _
    Chr(34) & "In Review" & Chr(34) & Chr(46) & m_s2Crs & _
    "The Reviewer must " & Chr(34) & "Reject" & Chr(34) & "before you " & _
    "can edit and re-submit your data."
End Sub
'================================================================================
Public Sub SetBadEntityCodeMsg()
m_sTitle = "Invalid Entity Code"
m_sPrompt = "Please enter a valid Entity Code." & m_s2Crs & _
    "NOTE:  Until you have done so you will not be allowed " & _
    "to save this workbook."
End Sub
'================================================================================
Public Sub SetCannotSubmitPostApprovalMsg()
m_sTitle = "Workbook Already Approved"
m_sPrompt = _
    "You cannot Submit Data after your work has been Approved by " & _
    "the Reviewer."
End Sub
'================================================================================
Public Sub SetCannotConnectToCTXOfficeMsg()
m_sTitle = "Problems Connecting to CorpTax Server"
m_sPrompt = _
    "We are experiencing problems connection with the CorpTax server.  " & _
```

```
    "You may want to try exiting completely out of Excel, or simply " & _
    "waiting a little while to see if either approach resolves this " & _
    "issue." & m_s2Crs & _
    "If the problem persists contact the Tax Technology group."
End Sub
'================================================================================
Public Sub SetWillYouBeReviewerQues()
m_sTitle = "Are You the Reviewer?"
m_sPrompt = _
    "Currently there is no reviewer assigned to this file." & m_s2Crs & _
    "Are you going to be the Reviewer?"
End Sub
'================================================================================
Public Sub SetReadOnlyAccessMsg()
m_sTitle = "Read-Only Access"
m_sPrompt = _
    "Since you are not the reviewer you will have read-only access " & _
    "to this file."
End Sub
'================================================================================
Public Sub SetEntCodeNotInSPMsg()
m_sTitle = "Entity Code Not Found"
m_sPrompt = _
    "The entity code you have entered is not in SharePoint." & m_s2Crs & _
    "Please verify your entry.  If your entry IS correct contact " & _
    "submit an Issue Tracker to have the SharePoint site updated." & m_s2Crs & _
    "NOTE:  You will not be able to save this workbook until this issue " & _
    "is resolved."
End Sub
'================================================================================
Public Sub SetInvalidTaxYearMsg(ByVal TaxPeriodList As String)
m_sTitle = "Invalid Period"
m_sPrompt = _
    "Please select one of the following Tax Years: " & m_s2Crs & _
    TaxPeriodList & m_s2Crs & _
    "[For the moment we are setting your tax year to " & Year(Now()) - 1 & ".]"
End Sub
'================================================================================
Public Sub SetInvalidDateMsg()
m_sTitle = "Invalid Entry"
m_sPrompt = "Please enter a valid date."
End Sub
'================================================================================
Public Sub SetChangingToValidYearMsg(ByVal YearEntered As Integer, _
    ByVal TaxYear As Integer)
m_sTitle = "Invalid Year"
m_sPrompt = "Your entry occurs in " & YearEntered & "." & m_s2Crs & _
    "We are changing your entry to " & TaxYear & "."
End Sub
'================================================================================
Public Sub SetUpdatingEndOfPeriodMsg(ByVal NewTaxYr As Integer)
m_sTitle = "Updating Period End"
m_sPrompt = "We are updating the Period End date to correspond to the new " & _
    "Tax Period (i.e., " & NewTaxYr & ")."
End Sub
'================================================================================
Public Sub SetCorrectingCTXCodeAndSnameEntryMsg(ByVal CorrectEntry As String)
'NOTE:  chr(34) = quotation mark (")
m_sTitle = "Incorrect SharePoint Property"
m_sPrompt = "The selection for " & Chr(34) & "Entity and Short Name" & _
Chr(34) & " seems to have been changed and no longer corresponds to the " & _
"entity you are working on in this file." & m_s2Crs & _
"This file will be saved with the SharePoint property set to:" & m_s2Crs1Tab & _
CorrectEntry
End Sub
'================================================================================
Public Sub SetSettingDocumentTitleMsg(ByVal TitleAlreadyExists As Boolean, _
    ByVal NewTitle As String)
Const sCREATING  As String = "creating"
Const sREPLACING  As String = "updating"
```

```
MsgBoxHandler - 6

Dim sCreateVsRpl As String
If TitleAlreadyExists Then
    sCreateVsRpl = sREPLACING
Else
    sCreateVsRpl = sCREATING
End If

m_sTitle = "Setting Document Title"
m_sPrompt = _
    "We are " & sCreateVsRpl & " the title of this document " & _
    "as:" & m_s2Crs1Tab & _
    NewTitle & m_s2Crs & _
    "Though we suggest you adhere to some variant of this naming " & _
    "convention you are free to change this title." & m_s3Crs & _
    "[NOTE:  If the SharePoint properties for this document are not " & _
    "visible you can access them by clicking on the Office button " & _
    "(at the top left of your screen) and selecting Prepare/Properties.]"
End Sub
'================================================================================
Public Sub SetDocTitleNotChangedMsg(ByVal OrigTitle As String)
m_sTitle = "No Change Made"
m_sPrompt = _
    "The document title has NOT been changed.  It remains:" & m_s2Crs1Tab & _
    OrigTitle
End Sub
'================================================================================
Public Sub SetCorrectingSPPptyMsg(ByVal SPPpty As String, _
    ByVal CorrectPptyVal As String)
m_sTitle = "Correcting SharePoint Property"
m_sPrompt = "You have an incorrect entry in the following " & _
    "SharePoint property:" & m_s2Crs1Tab & SPPpty & m_s2Crs & _
    "We wanted to let you know that we are setting the value of " & _
    "this property to the following:" & m_s2Crs1Tab & CorrectPptyVal
End Sub
'================================================================================
Public Sub SetSICodeNotFoundInSPMsg()
m_sTitle = "SharePoint Problem"
m_sPrompt = _
    "There is a problem with SharePoint.  Although you have entered a " & _
    "valid CorpTax/Entity code the corresponding SI code has not yet been " & _
    "set up on SharePoint." & m_s2Crs & _
    "Please inform Tax Technology of this issue and it will be " & _
    "resolved promptly."
End Sub
'================================================================================
Public Sub SetCorrectingSPPptsMsg(ByVal CorrectionList As String)
'NOTE:  chr(34) = quotation mark (")
m_sTitle = "Incorrect SharePoint Properties"
m_sPrompt = "At least one SharePoint property seems to have changed " & _
"and no longer corresponds to the entries in this Excel file." & m_s2Crs & _
"Accordingly, we are updating your SharePoint properties as follows:" & _
m_s2Crs & CorrectionList
End Sub
'================================================================================
Public Sub SetRunningOnPreXL07Msg()
m_sTitle = "Pre-2007 Office"
m_sPrompt = _
    "You are running on a version of Excel that is incompatible with " & _
    "the CorpTax Add-In.  " & m_s2Crs & _
    "Please close this workbook!"
End Sub
'================================================================================
Public Sub SetSaveDisabledDueToInvalidCTXCodeMsg()
m_sTitle = "Workbook Cannot Be Saved"
m_sPrompt = _
    "You will not be able to save this workbook until and unless you " & _
    "enter a valid CTX Entity Code."
End Sub
'================================================================================
Public Sub SetCannotGetUserNameMsg()
m_sTitle = "Cannot Get User Name"
```

```
m_sPrompt = _
    "In order to work properly FRED needs your user name.  " & _
    "However, this program is currently unable to obtain that datum." & _
    m_s2Crs & "Please report this issue in Issue Tracker.  " & _
    m_s2Crs & "In the meantime, this application will now close."
End Sub
'================================================================================
Public Sub SetYouHaveNotSavedWbkQues()
m_sTitle = "Workbook Not Saved... Try Again?"
m_sPrompt = _
    "The functionality of this worksheet will remain disabled until " & _
    "you save the changes (i.e., the Reviewer) to this file." & m_s2Crs & _
    "Would you like to save the changes after all?"
End Sub
'================================================================================
Public Sub SetReviewerUNApproveQues()
m_sTitle = "REVERSE Your Approval?"
m_sPrompt = _
    "You are about to reverse your prior approval.  As you know, this is " & _
    "a significant action on your part." & m_s3Crs & _
    "Are you CERTAIN that you want to REVERSE your approval?"
End Sub
'================================================================================
Public Sub SetWbkAlreadyApprovedMsg()
m_sTitle = "Workbook Already Approved"
m_sPrompt = _
    "This workbook has already been Approved and cannot " & _
    "be modified."
End Sub


'**************************************
'DEVELOPER MESSAGES
'**************************************
'================================================================================
Public Sub vSetRunDebugModeQues()
m_sTitle = m_sDEV_MSGBOX_TTL
m_sPrompt = "Run in Debug Mode?"
End Sub
'================================================================================
Public Sub vSetStepThruModeQues()
m_sTitle = m_sDEV_MSGBOX_TTL
m_sPrompt = "Execute in Step-Through mode?"
End Sub
'================================================================================
Public Sub vOverrideNotAReviewerQues()
m_sTitle = m_sDEV_MSGBOX_TTL
m_sPrompt = "CTX says you are not a reviewer.  Proceed as if you are one?"
End Sub
'================================================================================
Public Sub vSetStepThruWsChngQues()
m_sTitle = m_sDEV_MSGBOX_TTL
m_sPrompt = "Step through worksheet_change event?"
End Sub
'================================================================================
Public Sub vSetOmitPrintAreaTitlesQues()
m_sTitle = m_sDEV_MSGBOX_TTL
m_sPrompt = "OMIT Print_Titles and Print_Area names?"
End Sub
'================================================================================
Public Sub vSetBeAReviewerQues()
m_sTitle = m_sDEV_MSGBOX_TTL
m_sPrompt = "Enable Reviewer mode?"
End Sub
'================================================================================
Public Sub vSetBeASuperReviewerQues()
m_sTitle = m_sDEV_MSGBOX_TTL
m_sPrompt = "Enable SUPER-Reviewer mode?"
End Sub
'================================================================================
Public Sub vSetIgnoreDiffPreparerTrapQues()
```

```
m_sTitle = m_sDEV_MSGBOX_TTL
m_sPrompt = _
    "Enable Preparer Mode, even though you are not the Preparer" & _
    "(i.e., preparer is John Doe)?"
End Sub
'================================================================================
Public Sub vSetPrepForShutDownCompletedMsg()
m_sTitle = m_sDEV_MSGBOX_TTL
m_sPrompt = _
    "Done prepping for shut-down."
End Sub
'================================================================================
Public Sub vSetUpdateVersionDateOnDashboardQues()
m_sTitle = m_sDEV_MSGBOX_TTL
m_sPrompt = _
    "Update the version/date on our Dashboard Wsh?"
End Sub
'================================================================================
Public Sub vSetOmitPrintAreaTitlesMsg()
m_sTitle = m_sDEV_MSGBOX_TTL
m_sPrompt = "OMIT Print_Titles and Print_Area names?"
End Sub
'================================================================================
'================================================================================
'================================================================================
```