Sample code for SAP IBP Hooks to extend the functionality of the SAP IBP, add-in for Microsoft Excel

The SAP IBP, add-in for Microsoft Excel (Excel add-in) provides multiple hooks (entry points for custom coding) to enhance functions of the Excel add-in through Microsoft Visual Basic for Applications (VBA) code.

1. SAP IBP HOOKS FOR PLANNING VIEWS

It is possible to extend planning view workbooks with custom VBA implementations. The VBA code can be part of the planning view templates or can be called from a separate .xlam add-in (see the .xlam file provided in the folder together with this tutorial).

Remark: You might want to consider saving the VBA code to an .xlam add-in to be able to maintain it separately from the templates or favorites. Save the .xlam file on each user's PC at location **%appdata%\Microsoft\AddIns** and enable the custom Excel add-in by opening Microsoft Excel > File > Options > Add-ins > Manage > Go… selecting its name and confirming by choosing OK (see also following information provided by Microsoft Add or remove add-ins in Excel). The VBA code is then called according to the action that is taken by the user in the IBP Excel add-in.

The following hooks are available for planning views:

- IBPBeforeSend
- IBPAfterRefresh

1.1. IBPBeforeSend

The ***IBPBeforeSend*** hook is called when a user chooses ***Save Data*** or ***Simulate*** in the ***Data Input*** group or ***Create*** in the ***Scenarios*** group.

Your custom code runs right before the Excel add-in sends an update request to the back end. With this hook you might implement an extra validation step.

For further details about the parameters of the hook see IBPBeforeSend in the SAP Help Portal.

Sample Code for IBPBeforeSend:

The following code is a sample implementation which shows a message box, before saving changed key figure values. With this extra validation step, the user can stop the process by selecting "No" in the dialog box to avoid that the changes are saved persistently.

```
Function IBPBeforeSend(callMode As String) As Boolean

    'check which IBP action is currently in progress
    If callMode = "SAVE" Then

        'declare the variables
        Dim answer As Integer

        'show a message box
        answer = MsgBox(callMode & vbNewLine & "Some cells have been changed in worksheet '" & ActiveSheet.name & "'." & vbNewLine & "Do you want to proceed?", vbYesNo + vbQuestion, "Custom Validation via IBPBeforeSend")

        'stop or allow the process based on the decision made in the message box
        If answer = vbYes Then
            'Set the return value to True
            IBPBeforeSend = True
```

```
        Else
            'Set the return value to False
            IBPBeforeSend = False
        End If
    ElseIf callMode = "SIMULATE" Then
        'Set the return value to True
        IBPBeforeSend = True
    ElseIf callMode = "CREATE_SIMULATION" Then
        'Set the return value to True
        IBPBeforeSend = True
    End If
End Function
```

1.2. IBPAfterRefresh

The ***IBPAfterRefresh*** hook is called after the user chooses ***Refresh*** in the ***SAP IBP*** ribbon in the ***Data Input*** group. Your custom coding runs after the SAP IBP system has refreshed the data and the planning view is rewritten.

Note: ***IBPAfterRefresh*** is not a substitute of the hook ***AFTER_REFRESH***. It is called only after choosing Refresh, and not after, for example, choosing Save Data, Simulate, or after changing planning view settings.

For further details about the hook see IBPAfterRefresh in the SAP Help Portal.

Sample Code for IBPAfterRefresh:

With the following code a message comes up, each time after the planning view has been refreshed via choosing ***Refresh*** in the ***SAP IBP*** ribbon.

```
Function IBPAfterRefresh()
    MsgBox ("Your planning view has been successfully refreshed.")
End Function
```

2. SAP IBP HOOKS FOR MASTER DATA WORKBOOKS

It is possible to extend master data workbooks with custom VBA implementations. VBA hooks can be used for both master data and planning objects with key figure data worksheets.

To use the hooks for master data workbooks, the custom coding must be part of an .xlam add-in. When a master data workbook is saved as a favorite, the SAP IBP, add-in for Microsoft Excel only saves the configuration of the worksheets and generates the workbook on the fly when next time it is opened. The Microsoft Excel workbook binary is not saved with the master data favorites, which means that additional sheets and VBA code included are not kept.

The following hooks are available for master data workbooks:

- IBPMDAfterRefresh
- IBPMDBeforeUpdate

2.1. IBPMDAfterRefresh

The ***IBPMDAfterRefresh*** hook is called after opening a master data favorite, creating a new master data workbook or after choosing ***Refresh*** in the ***Master Data*** group of the ***SAP IBP*** ribbon.

Using this hook, you can manipulate the layout and appearance of master data worksheets, such as, for example, reordering, coloring, hiding columns.

For further details about the parameters of the hook see IBPMDAfterRefresh in the SAP Help Portal.

Sample code for IBPMDAfterRefresh:

The following code is a sample implementation which performs the following:

- If parameter ‘name’ equals ‘Product’ then
  - adds value help to ABC code if master data type ’Product’ is editable,
  - changes the cell color to blue for ‘Brand ID’,
  - arranges the Product ID and Product Desc columns next to each other.
- If parameter ‘name’ equals ‘Location’ then
  - hides GEOLATITUDE and GEOLONGITUDE attributes,
  - arrange the LOCID and LOCDESCR columns next to each other.

```
Function IBPMDAfterRefresh(name As String, editable As Boolean, rows As Long, headers() As String)

    'Declare the variables
    Dim idxWhat As Integer
    Dim idxWhere As Integer
    Dim addColumn As Integer
    Dim work As Range

    'If editable = true an additional column is added.
    If editable Then

        addColumn = 1

    Else

        addColumn = 0

    End If

    'If the opened worksheet includes the master data type "Product"
    If name = "Product" Then

        If editable Then

            'Add drop down value help to ABC code
            idxWhat = Application.Match("ABCID", headers, False) + addColumn

            Dim MyList(3) As String
            MyList(0) = "A"
            MyList(1) = "B"
            MyList(2) = "C"

            With Range(Cells(3, idxWhat), Cells(3 + rows + 30, idxWhat)).Validation
                .Delete
                .add Type:=xlValidateList, AlertStyle:=xlValidAlertStop, _
                Operator:=xlBetween, Formula1:=Join(MyList, ",")
            End With

        End If

        'Change color for Brand ID
        idxWhat = Application.Match("BRAND", headers, False) + addColumn
        Set work = Range(Cells(3, idxWhat), Cells(3 + rows + 30, idxWhat))
        work.Font.ColorIndex = 2
        work.Interior.ColorIndex = 5

        'Arrange the Product ID and Product Description columns next to each other
        idxWhere = Application.Match("PRDID", headers, False) + 1 + addColumn
        idxWhat = Application.Match("PRDDESCR", headers, False) + addColumn
        Columns(idxWhat).Cut
        Columns(idxWhere).Insert Shift:=xlToRight
```

```
    End If

    'If the opened worksheet includes the master data type "Location"
    If name = "Location" Then

        'Hide not necessary columns
        idxWhat = Application.Match("GEOLATITUDE", headers, False) + addColumn
        Columns(idxWhat).Hidden = True
        idxWhat = Application.Match("GEOLONGITUDE", headers, False) + addColumn
        Columns(idxWhat).Hidden = True

        'Arrange the Location ID and Location Description columns next to each other
        idxWhere = Application.Match("LOCID", headers, False) + 1 + addColumn
        idxWhat = Application.Match("LOCDESCR", headers, False) + addColumn
        Columns(idxWhat).Cut
        Columns(idxWhere).Insert Shift:=xlToRight

    End If
End Function
```

2.2. IBPMDBeforeUpdate

The ***IBPMDBeforeUpdate*** hook is called before the changes in the master data worksheet are sent to the back end after the user has chosen ***Save Changes…*** in the ***Master Data*** group of the ***SAP IBP*** ribbon.

It can be used, for example, to validate that naming conventions have been followed before saving data. It is also possible to cancel the save action by setting the IBPMDBeforeUpdate hook to False.

For further details about the parameters of the hook see IBPMDBeforeUpdate in the SAP Help Portal.

Sample code for IBPMDBeforeUpdate:

The following code is a sample implementation which validates the following attributes before saving master data changes.

- If the parameter ‘name’ equals ‘Product’ then
    - iterates the changedItems array and checks whether items contain text ‘IBP’
    - if an item contains text ‘IBP’ then shows a dialog box and sets the return value to False meaning the changes made by the user are not saved.
- If the parameter ‘name’ equals ‘Location’ then
    - shows a confirmation dialog box where the user can confirm the changes or stop the update process

```
Function IBPMDBeforeUpdate(name As String, headers() As String, changedItems() As String) As Boolean

    'declare the variables
    Dim answer As Integer
    Dim changeCount As Integer
    Dim productId As String
    Dim i As Integer

    'Set the return value to True
    IBPMDBeforeUpdate = True

    'If the opened worksheet includes the master data type "Product"
    If name = "Product" Then

        'do not allow that Product ID contains IBP
        changeCount = UBound(changedItems, 1)
        For i = 0 To changeCount
            productId = changedItems(i, 0)
```

```
                If productId Like "*IBP*" Then
                        IBPMDBeforeUpdate = False
                        MsgBox ("Please enter a Product ID, which does not include IBP.")
                        Exit Function
                End If
        Next i

    End If

    'If the opened worksheet includes the master data type "Location"
    If name = "Location" Then

        'Count the changes
        changeCount = UBound(changedItems, 1) + 1

        'Show a message box
        answer = MsgBox("Master Data:" & name & vbNewLine & changeCount & " rows has been
        modified." & vbNewLine & "Is custom validation OK?", vbYesNo + vbQuestion, "Custom
        Validation via IBPMDBeforeUpdate")

        'stop or allow the process based on the decision made in the message box
        If answer = vbYes Then
                'Set the return value to True
                IBPMDBeforeUpdate = True
        Else
                'Set the return value to False
                IBPMDBeforeUpdate = False
        End If

    End If
End Function
```

3. PERFORMANCE CONSIDERATIONS

Use the performance trace to check the runtime of the VBA code you implemented. For information about how to enable the performance trace, see SAP Note 2477564.

The local machine resources, the amount of data that is being processed and the VBA code implementation may impact the overall performance on the local computer. The performance trace can be enabled to analyze the execution time of the VBA hooks. If the execution time is longer, consider improving the implementation of the parts that are less performant and reduce the amount of data to be processed.

Keywords in the log files for the relevant VBA hooks are the following:

- IBPBeforeSend
  - Method [S&OP: IBPVBAHook (SAVE)]
  - Method [S&OP: IBPVBAHook (CREATE_SIMULATION)]
  - Method [S&OP: IBPVBAHook (SIMULATE)]
- IBPAfterRefresh
  - Method [S&OP: IBPVBAHook (REFRESH)]
- IBPMDAfterRefresh
  - Method [S&OP: IBPVBAHook (MASTERDATA_AFTER_REFRESH)]
- IBPMDBeforeUpdate
  - Method [S&OP: IBPVBAHook (MASTERDATA_BEFORE_SAVE)]

Follow us

www.sap.com/contactsap