PUBLIC

Prefill Reason Codes and Comments when Saving Data

Trigger Save Data with prefilling reason codes and comments and optionally suppressing the dialog for reason codes, comments and sharing. Additionally, the SAP IBP ribbon can be extended by using an .xml file and VBA coding included in a .xlam file.

HOW TO START

In other VBA use cases we already introduced the API SaveData. In this example, we show further use cases than just triggering save data the same way as if the user chooses “Save Data” in the ribbon of the SAP IBP, add-in for Microsoft Excel (Excel add-in).

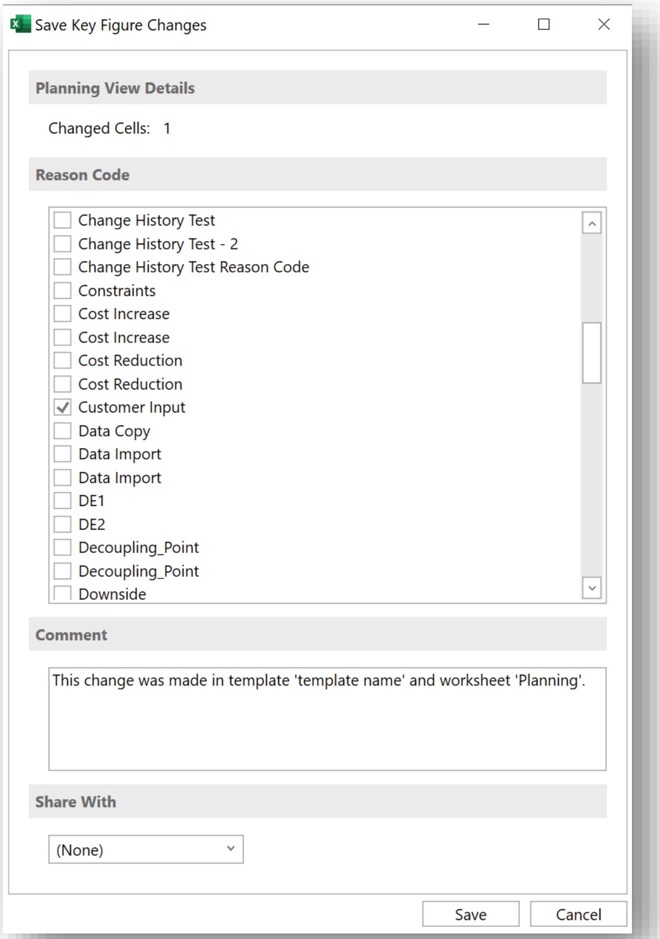
In this example, we show you how to prefill the dialog for reason codes, comments and sharing by using the API SaveData (see SaveData on the SAP Help Portal). You can preselect a specific reason code and you can enter a comment including the template name and the worksheet name where the changes which are to be saved have been made.

To make this function available for the user, you can include a VBA button in your templates with the following code:

```
Private IBPAutomationObject As Object

Private Sub SaveData_Click()
   Dim reasonCodes(1 To 1) As String
   Dim text As String
   Dim sheetName As String
   Dim templateName As String

On Error GoTo ErrorHandling:
   If IBPAutomationObject Is Nothing Then Set IBPAutomationObject =
   Application.COMAddIns("IBPXLClient.Connect").Object

   reasonCodes(1) = "Customer Input"
   sheetName = ActiveSheet.Name
   templateName = ActiveWorkbook.Names("SOP_Template_Name")
   'Cut off characters apart from the template name
   templateName = Right(templateName, Len(templateName) - 2)
   templateName = Left(templateName, Len(templateName) - 1)
   text = "This change was made in template '" & templateName & "' and worksheet '" & sheetName & "'."

   Call IBPAutomationObject.SaveData(False, reasonCodes, text)
   Exit Sub

ErrorHandling:
'Implement error handling to help the user to understand what went wrong
MsgBox Err.Description, vbOKOnly, "Microsoft Excel: Custom VBA code"

End Sub
```

If the user clicks the VBA button, after changing some key figure values, the dialog for reason codes, comments and sharing is shown with the prefilled reason code and comment (as shown in the screenshot below). The entries can be adjusted, and the changes can be saved by the user.

SAP Digital Supply Chain



Save Key Figure Changes

Planning View Details

Changed Cells: 1

Reason Code

- □ Change History Test
- □ Change History Test - 2
- □ Change History Test Reason Code
- □ Constraints
- □ Cost Increase
- □ Cost Increase
- □ Cost Reduction
- □ Cost Reduction
- ☑ Customer Input
- □ Data Copy
- □ Data Import
- □ Data Import
- □ DE1
- □ DE2
- □ Decoupling_Point
- □ Decoupling_Point
- □ Downside

Comment

This change was made in template 'template name' and worksheet 'Planning'.

Share With

(None)

Code explained row by row:

Please note: Code lines which already have been explained in tutorials of other use cases are not explained in detail again, so please check those as well.

```
Dim reasonCodes () As String
reasonCodes (1) = "reason code"
```

The parameter reasonCodes needs to be defined as string array. The entries may include the descriptions or IDs of the reason codes that are to be selected. Please note that only existing reason codes can be added, the values are case-sensitive.

```
templateName = ActiveWorkbook.Names("SOP_Template_Name")
```

Get the template name. For this, you can use the saved IBP-specific formula “SOP_Template_Name” (see Formulas > Defined Names > Name Manager). Even if an SAP IBP Favorite is created from the template, this name will stay untouched.

```
templateName = Right(templateName, Len(templateName) - 2)
templateName = Left(templateName, Len(templateName) - 1)
```

Cut the first two characters and the last one to get the template name.

```
sheetName = ActiveSheet.Name
```

Get the name of the active worksheet.

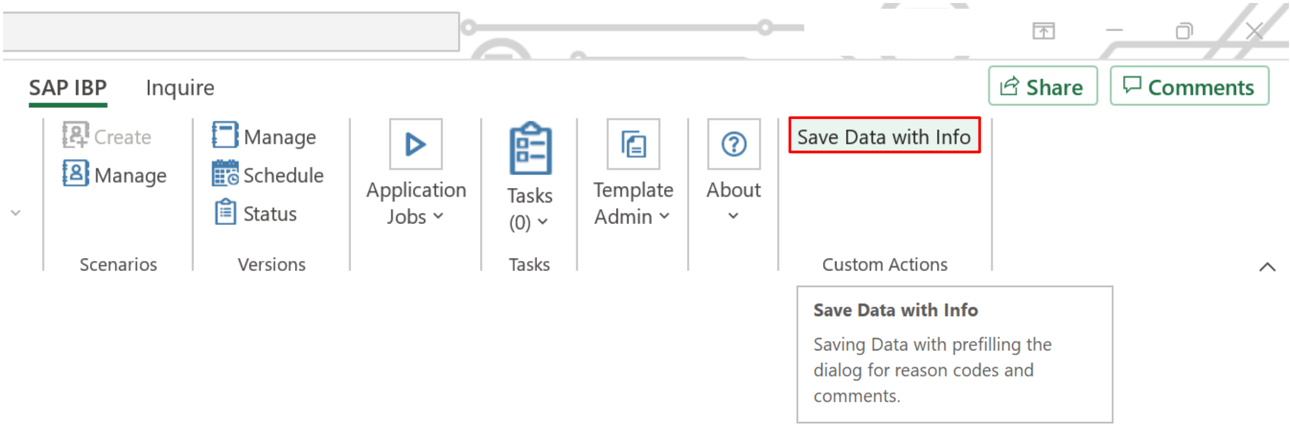
```
text = "This change was made in template '" & templateName & "' and worksheet '" & sheetName & "'"
```

Define the text you want to include as comment.

Call IBPAutomationObject.SaveData (False, reasonCodes, text)
Then call the API SaveData, and pass the parameters suppressReasonCodeCommentDialog, reasonCodes, and text. If the first parameter is set to “False” the dialog for reason codes, comments, and sharing is displayed and the user still needs to choose “Save” after checking the entries in the dialog.
If you set this parameter to “True”, the dialog will be suppressed, and the user doesn’t need to interact to save a reason code and a comment.
If you have the requirement that some of your users might need to enter reason codes while some don’t, you can include a VBA button for saving with suppressing the dialog only in specific templates.

Add the function to the SAP IBP Ribbon:

Another way to make this function available for all your users is to extend the SAP IBP ribbon by this function as shown below (see SAP IBP Ribbon Extension on the SAP Help Portal for further information).



Save Data with Info

Saving Data with prefilling the dialog for reason codes and comments.

To implement this, first create an .xml file with the following input and save it at location **%localappdata%\SAP_IBP_AddIn** with the name “**RibbonXML.xml**”.

```
<EPMTab xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://xml.sap.com/2010/02/bpc">
        <Group>
                <label>Custom Actions</label>
                        <Component>
                                <type>Button</type>
                                <label>Save Data with Info</label>
                                <supertip>Save Data with prefilling the dialog for reason codes and
                                comments </supertip>
                                <keytip></keytip>
                                <onAction>IBPExtension.xlam!SaveData_Click </onAction>
                                <isEnabled></isEnabled>
                                <onPressed></onPressed>
                        </Component>
        </Group>
</EPMTab>
```

Then create an .xlam file named “**IBPExtension.xlam**” with the same VBA code as shown above (SaveData_Click) and save it at location **%appdata%\Microsoft\AddIns**.
Enable the custom Excel add-in by opening Microsoft Excel > File > Options > Add-ins > Manage > Go… selecting “Ibpextension” and confirming by choosing OK (see also following information provided by Microsoft Add or remove add-ins in Excel.)

This needs to be done on each PC where the SAP IBP, add-in for Microsoft Excel should be extended with this function.
Both files (.xls and .xlam file) are provided in the same GitHub folder, where you have found this tutorial.

Follow us

www.sap.com/contactsap