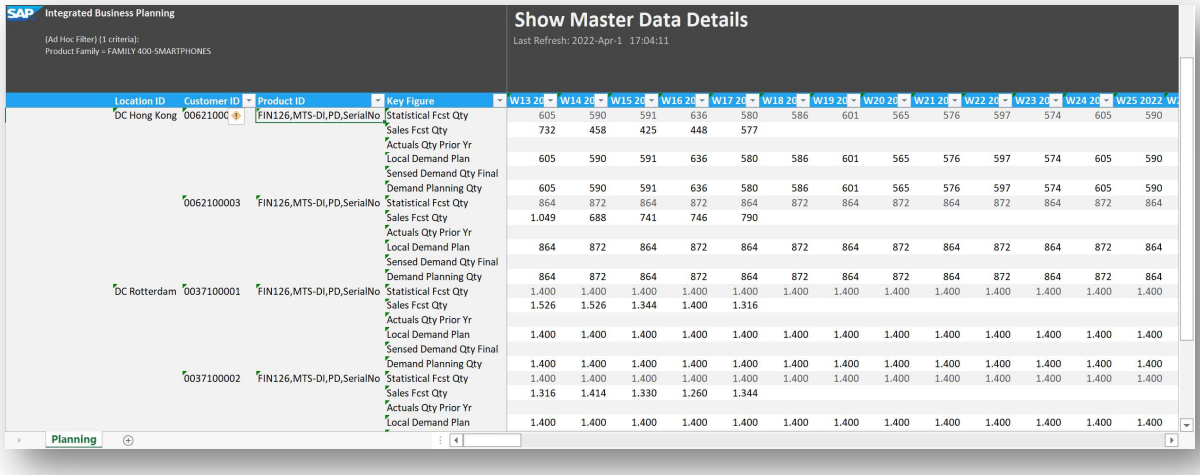
Show Master Data Details

With the newly provided API GetSingleMasterData, it is possible to retrieve the attribute values of a specific master data record by using VBA code. When double-clicking an attribute value in the planning view, a logic can be implemented to show further details about the attribute values of the specific master data record. As of today, business users would need to use the function ***Manage*** > ***Single...*** in the ***Master Data*** group in the ***SAP IBP*** ribbon or open a master data workbook to achieve the same result.

In this example, we show you how you can display master data details for specific product IDs in the planning view. The same code can be used and adjusted for other attributes.

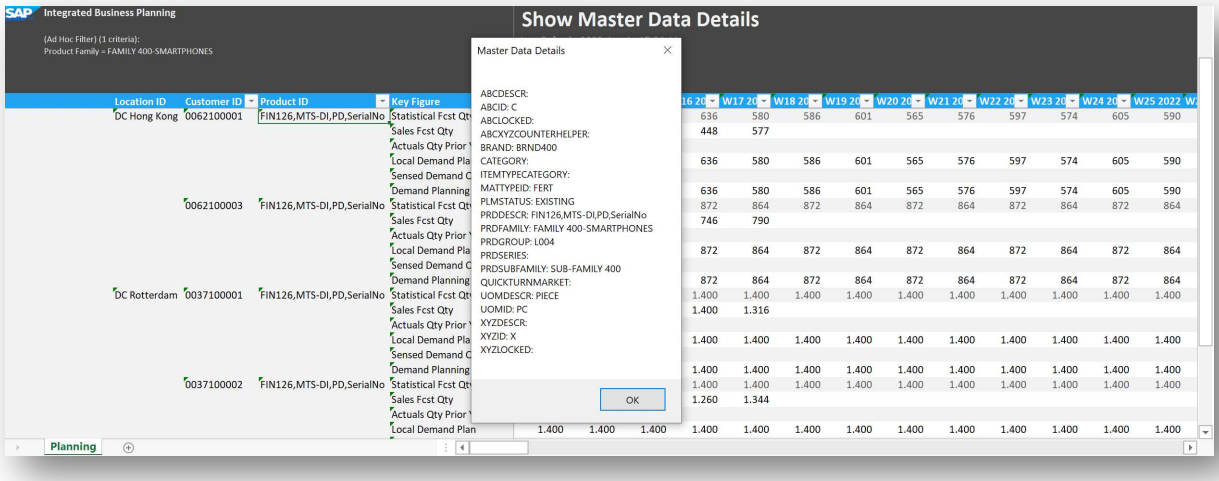
HOW TO START

As a first step, you need to create a planning view for which you want to include a specific logic to show master data details. To use our sample code, Product ID needs to be included as an attribute, in addtion to others.



The trigger to show the master data details of a specific Product ID is a double-click on that cell. To implement this, we use the Worksheet_BeforeDoubleClick VBA event provided by Microsoft in combination with the SAP IBP APIs GetSingleMasterData.

If a cell is double-clicked, we check whether the cell includes the Product ID. If it does, we determine the Product ID from the formula of the cell. Then we retrieve the attribute values of the Product ID from the master data type ***Product*** and show them in a message box (see screenshot below).



The code for this use case is part of the worksheet “Planning” and looks like the following:

```
Private IBPAutomationObject As Object

Private Sub Worksheet_BeforeDoubleClick(ByVal Target As Range, Cancel As Boolean)

Dim formula As String
Dim productID As String
Dim attributes() As String
Dim length As Integer
Dim details() As String
Dim attributeID As String
Dim attributeValue As String

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

   'Check if the cell includes the product ID
   formula = Target.Formula2
   If InStr(1, formula, "=@ EPMOlapMemberO(""[PRDID].[].[") = 0 Then
      Exit Sub
   End If

   'Determine the Product ID
   productID = Replace(formula, "=@ EPMOlapMemberO(""", "")
   productID = Left(productID, InStr(2, productID, "]"""))

   'Retrieve the attribute values of the Product ID from the master data type "Product"
   attributes = IBPAutomationObject.GetSingleMasterData("Product", Array(productID))

   'Determine the length of the array
   length = UBound(attributes)
   ReDim details(length)

   'Determine the ID and the value of the retrieved attributes
   For i = 0 To length
      attributeID = GetID(attributes(i))
      attributeValue = GetValue(attributes(i))
      details(i) = attributeID + ": " + attributeValue
```

```
    Next

    'Show a message box with the attribute ID and values of the master data record
    MsgBox Join(details, vbNewLine), vbOKOnly, "Master Data Details"

    Exit Sub

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

End Sub

Private Function GetID(attributeDef As String) As String
    GetID = Mid(attributeDef, 2, InStr(attributeDef, "].[].[") - 2)
End Function

Private Function GetValue(attributeDef As String) As String
    GetValue = Mid(attributeDef, InStr(attributeDef, "].[].[") + 6)
    GetValue = Left(GetValue, Len(GetValue) - 1)
End Function
```

Code explained row by row:

Please note: code lines which already have been explained in the tutorial for use cases provided with release 2202 are not explained in detail again, so please check those samples as well.

```
formula = Target.Formula2
If InStr(1, formula, "=@ EPMOlapMemberO(""[PRDID].[].[") = 0 Then
    Exit Sub
End If
```

Check whether the formula of the target cell which was double-clicked contains Product ID.

```
productID = Replace(formula, "=@ EPMOlapMemberO(""", "")
productID = Left(productID, InStr(2, productID, "]"""))
```

From the formula of the cell, filter out Product ID, which is needed as input parameter to call the API GetSingleMasterData. In our example we filter out [PRDID].[].[FG126]:

```
=@ EPMOlapMemberO("[PRDID].[].[FG126]";"";"FIN126,MTS-DI,PD,SerialNo";"";"000")
```

```
attributes = IBPAutomationObject.GetSingleMasterData("Product", Array(productID))
```

Call the API GetSingleMasterData, passing “Product” as masterDataType and Product ID as keyValues. The parameter keyValues is of type String[]. Therefore, you need to pass a one-dimensional string array, even if only one key attribute value is passed, see GetSingleMasterData on the SAP Help Portal for further information.

```
length = UBound(attributes)
```

Determine the length of the array which was retrieved.

```
ReDim details(length)
```

Use the ReDim statement provided by Microsoft to change the dimension of the array details().

```
For i = 0 To length
    attributeID = GetID(attributes(i))
    attributeValue = GetValue(attributes(i))
    details(i) = attributeID + ": " + attributeValue
 Next
```

The return value of the API GetSingleMasterData is a one-dimensional String array and contains the attribute values in following format: [attribute ID].[].[attribute value]. To not show the attributes in this technical format, you can determine the attribute ID and the attribute value by using the for loop and the

custom functions GetID and GetValue. Then, save the attribute ID and the attribute value in the right format to the details array.

```
MsgBox Join(details, vbNewLine), vbOKOnly, "Master Data Details"
```

Show the attribute IDs and attribute values as Msgbox in the format that you saved to the details array by using the Join function provided by Microsoft, and including a line break after each attribute. Display OK button only and insert “Master Data Details” as title.

```
Private Function GetID(attributeDef As String) As String
    GetID = Mid(attributeDef, 2, InStr(attributeDef, "].[].[") - 2)
End Function
```

This function is used to filter out the attribute ID from the format [attribute ID].[].[attribute value].

```
Private Function GetValue(attributeDef As String) As String
    GetValue = Mid(attributeDef, InStr(attributeDef, "].[].[") + 6)
    GetValue = Left(GetValue, Len(GetValue) - 1)
End Function
```

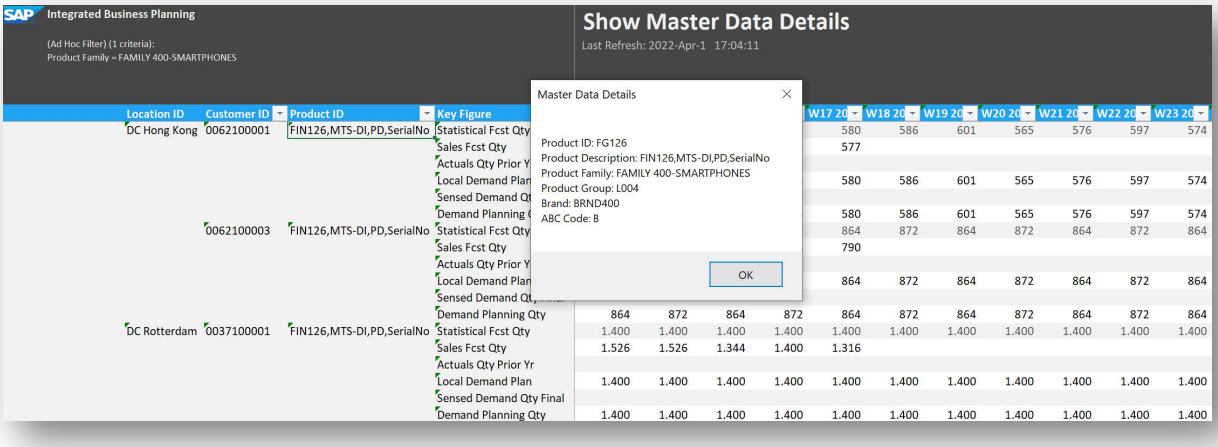
This function is used to filter out the attribute value from the format [attribute ID].[].[attribute value].

Further comments to the code:

- In this example, all related attributes of the master data type “Product” are displayed. If only specific attributes are relevant for your business users, you may filter them out from the attributes array, reorder them and only add the needed ones to the details array instead of adding one after the other in the same order. To do so, you can use the Filter function (VBA function provided by Microsoft) and a helper array. In the example code below, we filter out PRDDESCR (Product Description) from the attributes array, determine the ID and the value, and add them to the details array.

```
Dim helper ()  As String
helper = Filter(attributes, "[PRDDESCR].[].")
attributeID = GetID(helper(0))
attributeValue = GetValue(helper(0))
details(0) = attributeID + ": " + attributeValue
```

  The same can be done for other attributes (PRDFAMILY, PRDGROUP, BRAND, ABCID,…), and then exchange the part 'Determine the ID and the value’ of the retrieved attributes accordingly.



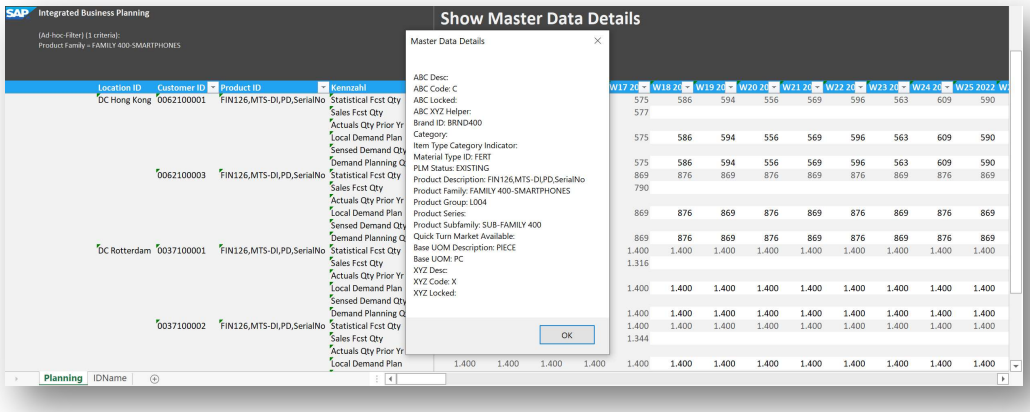
- If you want to display the attributes names and not the attribute IDs in the master data details dialog box, as of now there is no simple way to do so using an API. One possibility is to hard code the names in your code (as you see it in the screenshot above), or you may include the attribute IDs and their names in a separate worksheet, and determine the name of a specific ID by a VBA method (see the function GetName below).

	A	B
1	Attribute ID	Attribute Name
2	ABCDESCR	ABC Desc
3	ABCID	ABC Code
4	ABCLOCKED	ABC Locked
5	ABCXYZCOUNTERHELPER	ABC XYZ Helper
6	BRAND	Brand ID
7	CATEGORY	Category
8	ITEMTYPECATEGORY	Item Type Category Indicator
9	MATTYPEID	Material Type ID
10	PLMSTATUS	PLM Status
11	PRDDESCR	Product Description
12	PRDFAMILY	Product Family
13	PRDGROUP	Product Group
14	PRDSERIES	Product Series
15	PRDSUBFAMILY	Product Subfamily
16	QUICKTURNMARKET	Quick Turn Market Available
17	UOMDESCR	Base UOM Description
18	UOMID	Base UOM
19	XYZDESCR	XYZ Desc
20	XYZID	XYZ Code
21	XYZLOCKED	XYZ Locked
22		
23		
24		
25		
26		

Planning | IDName

The part in the code marked in yellow would need to be changed as follows, and the function GetName needs to be added (see xlsm file “Show Master Data Details_WithName”):

```
'Determine the ID and the value of the retrieved attributes
For i = 0 To length
    attributeID = GetID(attributes(i))
    attributeName = GetName(attributeID)
    attributeValue = GetValue(attributes(i))
    details(i) = attributeName + ": " + attributeValue
Next
Private Function GetName(attributeID As String) As String
  Dim position As range
  Set position = Worksheets("IDName").range("A2:A21").Find(attributeID, LookIn:=xlValues)
  GetName = Worksheets("IDName").Cells(position.Row, 2).Text
End Function
```



So as not to confuse the business user, the additional worksheet “IDName” can be hidden.

- If you want to implement a generic logic that every attribute can be double-clicked, and additional master data information is displayed, some more effort and VBA code is needed. To retrieve master data attribute values using the API GetSingleMasterData, the ID or name of the master data type and the key attribute values are needed.
  - The master data type is hard coded in our sample, and works with all planning areas where the name of the master data type PRODUCT equals “Product”. To retrieve the complete list of master data type IDs of the planning area you are connected to, you can use the API GetMasterDataTypes. To then determine the master data type of a specific attribute, there is currently no easy solution, so as shown in the example above, for attribute ID and attribute name you would need to create a worksheet where the dependencies between attributes and master data types are listed, which can be used in the VBA code.
  - If you already have the master data type, you can determine the ID of its key attributes by using the API GetMasterDataTypeKeyAttributes. To then determine the key attribute values the easiest way is to have the key attributes included in the planning view. You can then determine the key attribute values by using the API GetAttributeValues (see further details how this can be implemented in our sample ***1.3 Change Master Data Attribute Value***).

  Find further information about the different APIs which are available on the SAP Help Portal at SAP IBP APIs.

Follow us

www.sap.com/contactsap