

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, the 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.

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 also 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, the Product ID needs to be included as attribute, a part of others.

Location ID	Customer ID	Product ID	Key Figure	W13 20	W14 20	W15 20	W16 20	W17 20	W18 20	W19 20	W20 20	W21 20	W22 20	W23 20	W24 20	W25 2022
DC Hong Kong	006210003	FIN126.MTS-DI.PD.SerialNo	Statistical Fcst Qty	605	590	591	636	580	586	601	565	576	597	574	605	590
			Sales Fcst Qty	732	458	425	448	577								
			Actuals Qty Prior Yr													
			Local Demand Plan	605	590	591	636	580	586	601	565	576	597	574	605	590
			Sensed Demand Qty Final	605	590	591	636	580	586	601	565	576	597	574	605	590
			Demand Planning Qty	605	590	591	636	580	586	601	565	576	597	574	605	590
			Statistical Fcst Qty	864	872	864	872	864	872	864	872	864	872	864	872	864
			Sales Fcst Qty	1.049	688	741	746	790								
			Actuals Qty Prior Yr													
			Local Demand Plan	864	872	864	872	864	872	864	872	864	872	864	872	864
			Sensed Demand Qty Final	864	872	864	872	864	872	864	872	864	872	864	872	864
			Demand Planning Qty	864	872	864	872	864	872	864	872	864	872	864	872	864
			Statistical Fcst Qty	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400
			Sales Fcst Qty	1.526	1.526	1.344	1.400	1.316								
			Actuals Qty Prior Yr													
			Local Demand Plan	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400
			Sensed Demand Qty Final	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400
			Demand Planning Qty	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400
			Statistical Fcst Qty	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400
			Sales Fcst Qty	1.316	1.414	1.330	1.260	1.344								
			Actuals Qty Prior Yr													
			Local Demand Plan	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400	1.400

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 yes, 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 screenshot shows the SAP IBP 'Show Master Data Details' dialog. The dialog lists various master data attributes for a selected product. The background table shows data for different locations and products, with columns for Location ID, Customer ID, Product ID, and Key Figure.

The code for this use case, is part of the worksheet "Planning" and looks 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 ErrorHandler:

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 the Product ID

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

From the formula of the cell, filter out the 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 the Product ID as keyValues. The parameter keyValues is from 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
```

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 may use the **Filter function** (VBA function provided by Microsoft) and a helper array. In the example code below we filter out the PRDDESCR (Product Description) from the attributes array, determine the ID and the value and add it 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.

The screenshot shows the SAP Integrated Business Planning interface. A 'Show Master Data Details' pop-up window is displayed over a table. The pop-up window contains the following information:

- Product ID: FG126
- Product Description: FIN126,MTS-DI,PD,SerialNo
- Product Family: FAMILY 400-SMARTPHONES
- Product Group: L004
- Brand: BRND400
- ABC Code: B

The background table has columns: Location ID, Customer ID, Product ID, and Key Figure. It shows data for two locations: DC Hong Kong and DC Rotterdam. The 'Key Figure' column lists various statistical and demand planning metrics.

- If you want to display the attributes names and not the attribute IDs in the master data details pop-up, as of now there is no simple way to do so via 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	ABLOCKED	ABC Locked
5	ABXYZCOUNTERHELPER	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		

The following part in the code (marked in yellow) would need to be changed as follows and the function GetName needs to be added (see xlsx 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

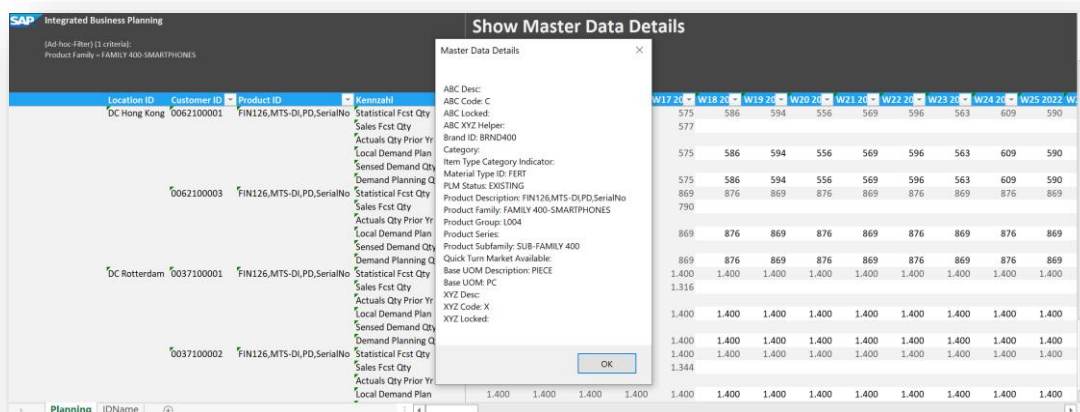
Private Function GetName(attributeID As String) As String

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



To not 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 will be displayed, some more effort and VBA code is needed. To retrieve master data attribute values via 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 is equals to “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

© 2022 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platforms, directions, and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

See www.sap.com/trademark for additional trademark information and notices.