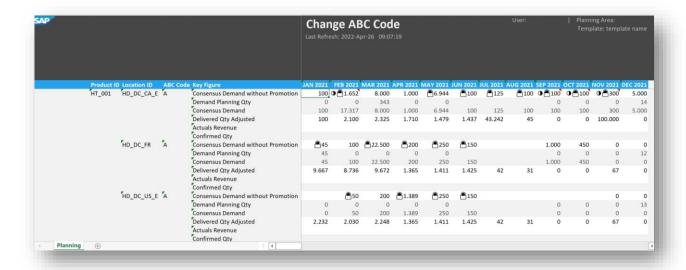
# **Change Master Data Attribute Value**

With the newly provided API UpdateSingleMasterData, it is possible to change the value of a specific master data attribute by using VBA code. If your business users must change one specific master data attribute several times a day, you might include a specific logic in your template to make it possible to change the attribute value without the need of choosing *Manage* > *Single...* in the *Master Data* group in the *SAP IBP* ribbon.

In our example, we explain how to implement changing the ABC code for a specific Product ID by double-clicking the cell of the ABC code in the planning view. The same code can be used for other attributes than ABC code as well, so as always feel free to adjust the code to your needs.

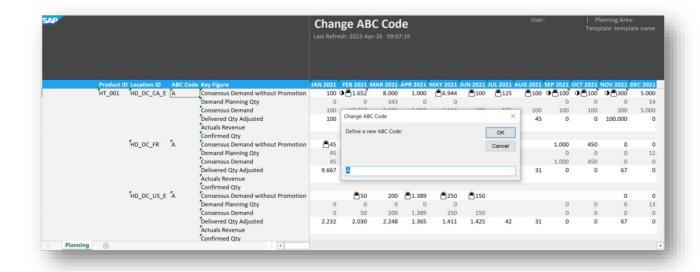
#### **HOW TO START**

As a first step, you need to create a planning view for which you want to include a specific logic to update master data attribute values. To use our sample code, the ABC Code needs to be selected as an attribute, in addition to others.



The trigger for the change of the ABC code is a double-click on the cell of the ABC code of a specific product ID. To implement this, we use the <a href="Worksheet\_BeforeDoubleClick">Worksheet\_BeforeDoubleClick</a> VBA event provided by Microsoft, in combination with the SAP IBP APIs GetAttributeValues and UpdateSingleMasterData.

If a cell is double-clicked, we check whether the cell includes the ABC code. If it does, we determine the Product ID of the cell which was double-clicked with the API GetAttributeValues. Then, we open an inputbox to allow the user to enter a new value for the ABC code.



Once the user confirmed the input by clicking ok, we are validating the input, and calling the API UpdateSingleMasterData, passing the master data type **Product**, the Product ID, and the new ABC code.

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 ABCcode As String Dim attributes() As String Dim formula Dim newABC As Variant

On Error GoTo ErrorHandling:

If IBPAutomationObject Is Nothing Then Set IBPAutomationObject = Application.COMAddIns("IBPXLClient.Connect").Object

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

'Determine the Product ID

attributes = IBPAutomationObject.GetAttributeValues(Target.Offset(0, 4))

attributes = Filter(attributes, "[PRDID].[].")

If UBound(attributes) < 0 Then

MsgBox "The Product ID couldn't be determined.", vbOKOnly, "Microsoft Excel: Custom VBA code" Exit Sub

End If

'Show a inputbox for entering a new ABC code

ABCcode = Target. Value2

newABC = InputBox("Define a new ABC Code:", "Change ABC Code", ABCcode)

'Check the input

If newABC = "A" Or newABC = "B" Or newABC = "C" Then

'Update ABCCode for the specific Product ID

ABCcode = "[ABCID].[].[" + newABC + "]"

Call IBPAutomationObject.UpdateSingleMasterData("Product", attributes, Array(ABCcode))

Call IBPAutomationObject.Refresh

Else

MsgBox "The input you entered is not valid.", vbOKOnly, "Microsoft Excel: Custom VBA code"

# SAP Digital Supply Chain

THE BEST RUN

End If 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

## Code explained row by row:

<u>Please note:</u> 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.

If IBPAutomationObject Is Nothing Then Set IBPAutomationObject = Application.COMAddIns("IBPXLClient.Connect").Object

If the SAP IBP automation object was not set yet, we retrieve it from the SAP IBP, add-in for Microsoft Excel (Excel add-in).

End If

Check whether the formula of the target cell which was double-clicked contains the ABC code.

#### attributes = IBPAutomationObject.GetAttributeValues(Target.Offset(0, 4))

Determine the attribute values of the target cell which was double-clicked.

As the range which is provided as parameter for GetAttributeValues needs to be part of the data area of the planning view, we need to shift the target range to the right by several columns. The exact number of shifted columns depends on the planning view that you have created (the number of attributes and where the ABC code column is situated).

# attributes = Filter(attributes, "[PRDID].[].")

Filter out the entry for Product ID from the array of attribute values, by using the <u>Filter function</u> (VBA function provided by Microsoft).

If UBound(attributes) < 0 Then

MsgBox "The Product ID couldn't be determined.", vbOKOnly, "Microsoft Excel: Custom VBA code" Exit Sub

End If

If the filtered array contains no entries, show the error message "The Product ID couldn't be determined." and exit the sub, else continue with the next steps.

## ABCcode = Target.Value2

Save the current value of the target cell which was double-clicked as ABCcode using the Property <u>Value2</u> provided by Microsoft.

newABC = InputBox("Define a new ABC Code:", "Change ABC Code", ABCcode)

Open an <u>inputbox</u> with prompt "Define a new ABC Code:", title "Change ABC Code" and default value ABCcode and save the entered value as newABC.

If newABC = "A" Or newABC = "B" Or newABC = "C" Then

Else

MsgBox "The input you entered is not valid.", vbOKOnly, "Microsoft Excel: Custom VBA code"

Check the input. In our case only "A", "B" or "C" are allowed, else the error message "The input you entered is not valid." shows up.

# ABCcode = "[ABCID].[].[" + newABC + "]"

To pass the non-key attribute values that you want to change (parameter nonKeyValues of API UpdateSingeMasterData), the attribute values need to have the format [attribute ID].[].[attribute value] in this case for example [ABCID].[].[A].

# **SAP Digital Supply Chain**

THE BEST RUN

# Call IBPAutomationObject.UpdateSingleMasterData("Product", attributes, Array(ABCcode))

Call the API UpdateSingleMasterData, passing "Product" as masterDataType, the attributes array with the Product ID as keyValues and Array(ABCCode) as nonKeyValues.

Both parameters keyValues and nonKeyValues are of type String[]. Therefore, you need to pass a one-dimensional string array, even if only one key attribute value or non-key attribute value is passed. Please make sure to only pass non-key attribute values that are different from the actual saved value, else an error message will be shown. See <a href="UpdateSingleMasterData">UpdateSingleMasterData</a> on the SAP Help Portal for further information.

# Call IBPAutomationObject.Refresh

Call the API Refresh to refresh the active worksheet. With this, the changed value will become visible for the user in the planning view.

#### Further comments to the code:

- Adjust the offset of the target cell to be used for GetAttributeValues according to your planning view.
- If you want to implement a generic logic that every attribute can be double-clicked and changed, some
  more effort and VBA code is needed. Similar applies as already described in sample "1.2 Show Master
  Data Details". To change master data attribute values using the API UpdateSingle-MasterData, the ID or
  name of the master data type and the key attribute values are needed. Please check the other tutorial for
  further information/ideas.
- The API UpdateSingleMasterData is not meant for mass changes of master data attributes. If you want to change a specific attribute value as shown in our example this is a good option.

#### 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.