Concatenation of Long Text in BEx Workbooks

Applies to:

SAP BI 7.0. For more information, visit the Business Intelligence homepage.

Summary

This article explains on adding better functionality and front end formatting to BEx workbooks using Excel VBA (Macro). This piece of VBA code is used to concatenate text from many columns in a workbook and display it in a single column to the user.

Author: Ghuru Balaji

Company: Wipro Technologies

Created on: 21 August 2009

Author Bio

Ghuru Balaji PS is a BI Consultant for Wipro Technologies

Table of Contents

Basics

Macro is basically VBA code which can be used in MS Excel to perform different functions like

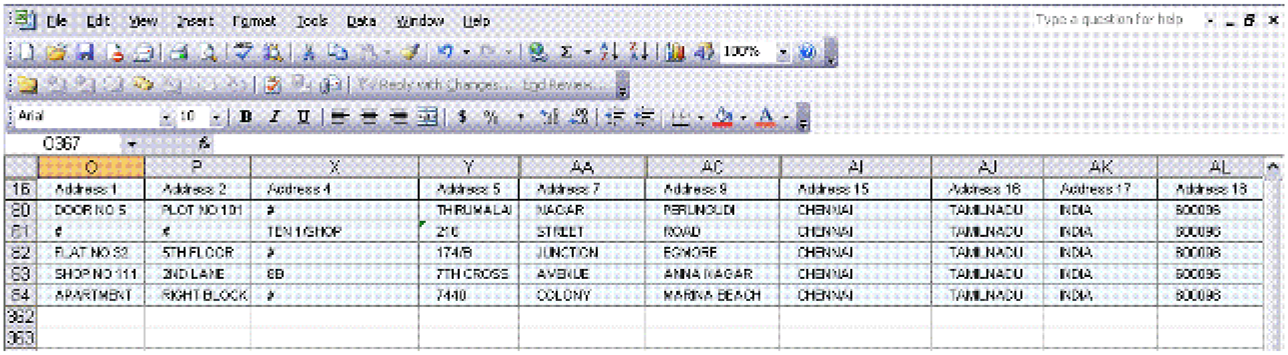
- Copying sheets within workbooks
- Copying data within workbooks etc

In this article we will discuss about combining text from many columns and displaying in a single column.

Drill down and drill across are supported after implementing this piece of code.

Concatenation Scenario:

Before concatenation:



File Edit View Insert Format Tools Data Window Help
Type a question for help
100%
Reply with Changes... End Review...
Arial
10
O367

	O	P	X	Y	AA	AC	AI	AJ	AK	AL
16	Address 1	Address 2	Address 4	Address 5	Address 7	Address 9	Address 15	Address 16	Address 17	Address 18
60	DOOR NO 5	PLOT NO 101	#	THIRUMALAI	NAGAR	PERUNGUDI	CHENNAI	TAMILNADU	INDIA	600096
61	#	#	TEN 1 SHOP	216	STREET	ROAD	CHENNAI	TAMILNADU	INDIA	600096
62	FLAT NO 32	5TH FLOOR	#	174/B	JUNCTION	EGMORE	CHENNAI	TAMILNADU	INDIA	600096
63	SHOP NO 111	2ND LANE	6B	7TH CROSS	AVENUE	ANNA NAGAR	CHENNAI	TAMILNADU	INDIA	600096
64	APARTMENT	RIGHT BLOCK	#	7440	COLONY	MARINA BEACH	CHENNAI	TAMILNADU	INDIA	600096
362										
363										

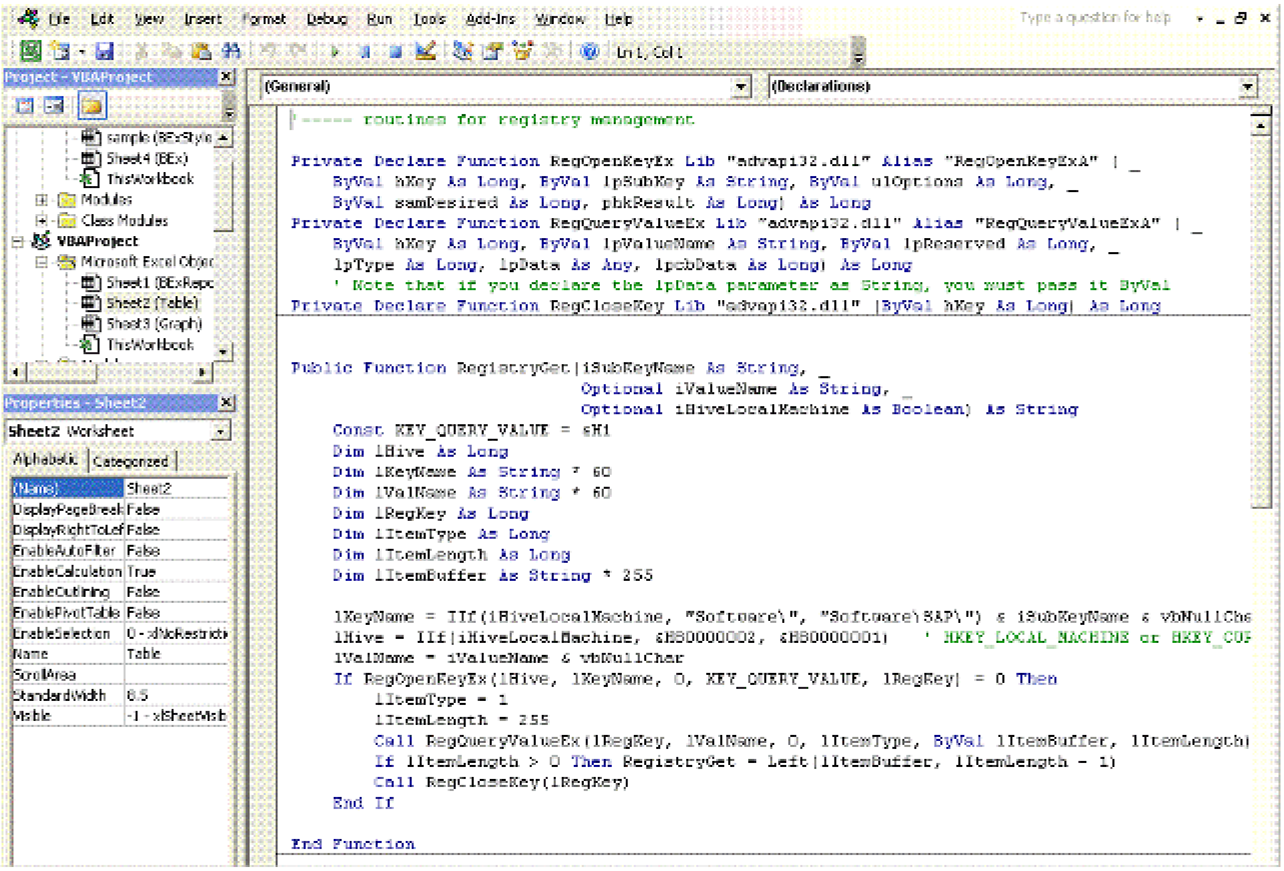
Using VBA in BEx

1. Navigation to Visual Basic Editor.

Open BEx Analyzer with any workbook (Sample workbook shown below) Navigate to

Tools -> Macro -> Visual Basic Editor

A new window opens up as shown below:



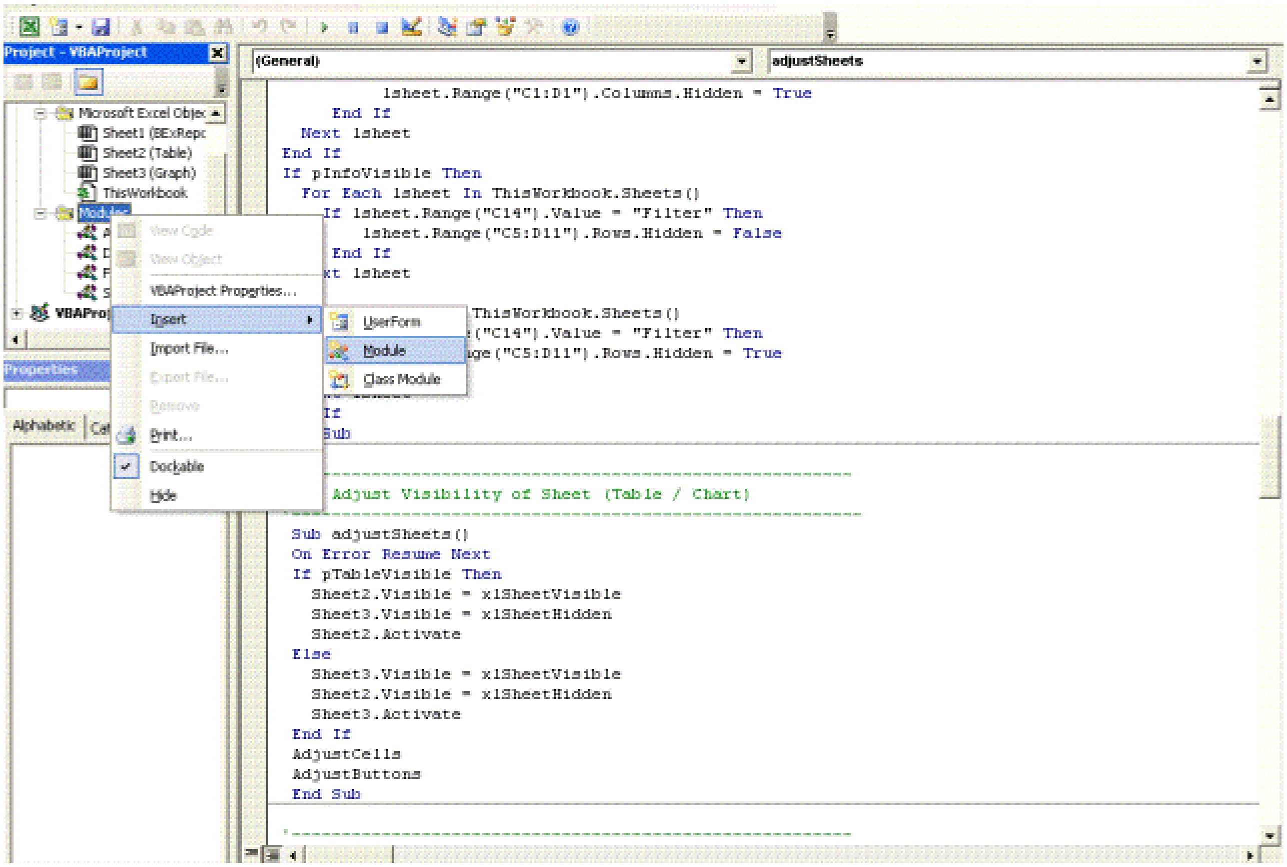
(Name)	Sheet2
DisplayPageBreak	False
DisplayRightToLef	False
EnableAutoFilter	False
EnableCalculation	True
EnableOutlining	False
EnablePivotTable	False
EnableSelection	0 - xlNoRestricti
Name	Table
ScrollArea	
StandardWidth	8.5
Visible	-1 - xlSheetVisib

```
'----- routines for registry management

Private Declare Function RegOpenKeyEx Lib "advapi32.dll" Alias "RegOpenKeyExA" ( _
    ByVal hKey As Long, ByVal lpSubKey As String, ByVal ulOptions As Long, _
    ByVal samDesired As Long, phkResult As Long) As Long
Private Declare Function RegQueryValueEx Lib "advapi32.dll" Alias "RegQueryValueExA" ( _
    ByVal hKey As Long, ByVal lpValueName As String, ByVal lpReserved As Long, _
    lpType As Long, lpData As Any, lpcbData As Long) As Long
    ' Note that if you declare the lpData parameter as String, you must pass it ByVal
Private Declare Function RegCloseKey Lib "advapi32.dll" (ByVal hKey As Long) As Long


Public Function RegistryGet(iSubKeyName As String, _
                            Optional iValueName As String, _
                            Optional iHiveLocalMachine As Boolean) As String
    Const KEY_QUERY_VALUE = &H1
    Dim lHive As Long
    Dim lKeyName As String * 60
    Dim lValName As String * 60
    Dim lRegKey As Long
    Dim lItemType As Long
    Dim lItemLength As Long
    Dim lItemBuffer As String * 255

    lKeyName = IIf(iHiveLocalMachine, "Software\", "Software\SAP\") & iSubKeyName & vbNullCha
    lHive = IIf(iHiveLocalMachine, &H80000002, &H80000001)    ' HKEY_LOCAL_MACHINE or HKEY_CUR
    lValName = iValueName & vbNullChar
    If RegOpenKeyEx(lHive, lKeyName, 0, KEY_QUERY_VALUE, lRegKey) = 0 Then
        lItemType = 1
        lItemLength = 255
        Call RegQueryValueEx(lRegKey, lValName, 0, lItemType, ByVal lItemBuffer, lItemLength)
        If lItemLength > 0 Then RegistryGet = Left(lItemBuffer, lItemLength - 1)
        Call RegCloseKey(lRegKey)
    End If

End Function
```

Note: The left side window shows your workbook name and details, also called the Project explorer window and the right side window gives the detailed descriptions.

2. Where to enter the code?

a. Right click on modules tab, choose insert -> module as shown below:



```
            lsheet.Range("C1:D1").Columns.Hidden = True
        End If
    Next lsheet
End If
If pInfoVisible Then
    For Each lsheet In ThisWorkbook.Sheets()
      If lsheet.Range("C14").Value = "Filter" Then
          lsheet.Range("C5:D11").Rows.Hidden = False
        End If
      xt lsheet

                    ThisWorkbook.Sheets()
                    ("C14").Value = "Filter" Then
                    ge("C5:D11").Rows.Hidden = True

      If
      Sub

      '---------------------------------------------------------
        Adjust Visibility of Sheet (Table / Chart)
      '---------------------------------------------------------
 Sub adjustSheets()
 On Error Resume Next
 If pTableVisible Then
   Sheet2.Visible = xlSheetVisible
   Sheet3.Visible = xlSheetHidden
   Sheet2.Activate
 Else
   Sheet3.Visible = xlSheetVisible
   Sheet2.Visible = xlSheetHidden
   Sheet3.Activate
 End If
 AdjustCells
 AdjustButtons
 End Sub

'---------------------------------------------------------
```

- Copy the given below code and paste it in the created new Module

```
      Sub entryVariable(findHeading As String, startDataRange As Integer, noOfColumns As Integer,
      sheet As String, noOfSheets As Integer)

    findHeading = " " 'Enter the Heading which needs to be searched
    noOfColumns = " " 'Enter number of Columns to be concatenated
    startDataRange = " " 'Enter Starting row of records
    sheet = " " 'Enter Sheet Names seperated by comma if you have multiple sheets
    noOfSheets = " " 'Enter Number of Sheets
                  End Sub
Sub Concatenation(rowCount As Integer, gridName As String)
Dim findHeading As String
Dim noOfColumns As Integer
Dim numberColumn As Integer
Dim numberRow As Integer
Dim numberStartColumn As Integer
Dim numberEndColumn As Integer
Dim startDataRange As Integer
Dim sheetName As String
Dim sheet As String
Dim noOfSheets As Integer
Dim noColumn As Integer
Dim Grid As Range
Dim startRow As Integer, endRow As Integer
Dim startCol As String, startColConcatenate As String, endColConcatenate As String
Dim startGrid As Integer, endGrid As Integer
' Call Sub Module to fetch all the entry variables
entryVariable findHeading, startDataRange, noOfColumns, sheet, noOfSheets
' Call Sub Module to find the sheet and grid name
splitSheetName sheet, gridName, sheetName, noOfSheets
' Call Sub Module to find the heading to be concatenated
Find findHeading, numberStartColumn, numberRow, sheetName
numberColumn = numberStartColumn - 1
numberEndColumn = numberColumn + noOfColumns
startRow = numberRow + 1
endRow = startDataRange + rowCount - 2
' Call Sub Module to find Column String
columnString numberColumn, startCol
columnString numberEndColumn, endColConcatenate
columnString numberStartColumn, startColConcatenate
' Select the columns to be hidden
Columns(startColConcatenate & ":" & endColConcatenate).Select
' Hide the columns
Selection.EntireColumn.Hidden = True
' Select the Column where the concatenation needs to take place
Columns(startCol & ":" & startCol).Select
' Assign Column Width of 50
Selection.ColumnWidth = 50
Range(startCol & startRow).Select
' Call Sub Module to Concatenate the address
address_Concatenate noOfColumns
' Execute the auto fill if the no of rows is not equal to startdatarange
If endRow <> startDataRange Then
```

```
Selection.AutoFill Destination:=Range(startCol & startRow & ":" & startCol & endRow),
Type:=xlFillDefault
End If
' Select the records which are concatenated
Range(startCol & startRow & ":" & startCol & endRow).Select
Selection.Copy
' Paste the Values without the formulas
Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
Selection.EntireColumn.AutoFit
Selection.Application.CutCopyMode = False
Range("F14").Select
End Sub
Sub Find(searchHeading As String, noColumn As Integer, noRow As Integer, sheetName As
String)
    'Select the sheet particular to the Grid
    Sheets(sheetName).Select
    'Find the Cell Which we need to search
    Cells.Find(What:=searchHeading, After:=ActiveCell, LookIn:=xlFormulas, LookAt _
    :=xlPart, SearchOrder:=xlByRows, SearchDirection:=xlNext, MatchCase:= _
    False, SearchFormat:=False).Activate
    'Assign the Column and Row Value
    noColumn = ActiveCell.Column
    noRow = ActiveCell.Row
End Sub
Sub columnString(noColumn As Integer, columnHeader As String)
' Find the Column from the Integer value of the Column
    If (noColumn / 26) > 1 And (noColumn / 26) < 2 Then
        columnHeader = Chr(65) & Chr(64 + (noColumn Mod 26))
    ElseIf (noColumn / 26) >= 2 And (noColumn / 26) < 3 Then
        columnHeader = Chr(66) & Chr(64 + (noColumn Mod 26))
    Else
       columnHeader = Chr(64 + noColumn)
    End If
End Sub
Sub address_Concatenate(noColumn As Integer)
'Address Concatenation
Dim Address As String
Dim replaceSpecialChar As String
Dim Count As Long
Dim countConcatenate As Long
[
Address = ""
        For countConcatenate = 1 To noColumn
            If countConcatenate = 1 Then
          Address = "=CONCATENATE(IF(RC[" & countConcatenate & "]=""#"","""",RC[" &
countConcatenate & "] & "" ""),"
            ElseIf countConcatenate = noColumn Then
           Address = Address & "IF(RC[" & countConcatenate & "]=""#"","""",RC[" & countConcatenate
& "]))"
            Else
           Address = Address & "IF(RC[" & countConcatenate & "]=""#"","""",RC[" & countConcatenate
& "] & "" ""),"
            End If
            Next countConcatenate
ActiveCell.FormulaR1C1 = Address
```

```
End Sub
Sub splitSheetName(sheet As String, gridName As String, sheetName As String, noOfSheets
As Integer)
'To find the sheet name
Dim gridNameSplit() As String
Dim sheetNameSplit() As String
Dim Count As Long
Dim sheetNumber As Long
    gridNameSplit = Split(gridName, "_") ' Find the Grid Name and assign it to an array
    sheetNameSplit = Split(sheet, ",") ' Find the sheet Name and assign it to an array
For Count = 1 To noOfSheets
If gridNameSplit(1) = Count Then
    sheetNumber = Count - 1
    sheetName = sheetNameSplit(sheetNumber) ' Assign the sheet name specific to the grid
End If
Next Count
End Sub
```

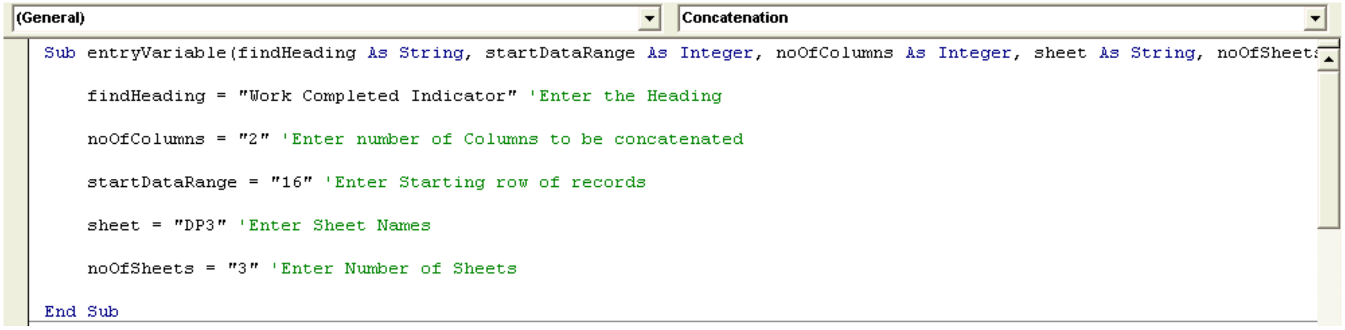
- Rename the Module as per your requirement

b. In the Project Explorer Window Choose ***DefaultWorkbook*** Module and insert the below Code in the ***CallBack*** Sub Function

```
Dim gridName As String
gridName = varname(2)
Module1.ConcatColumnsNew (varname(1).Rows.Count)
```

Note : *Module1* must be replaced with the name you had given for the module

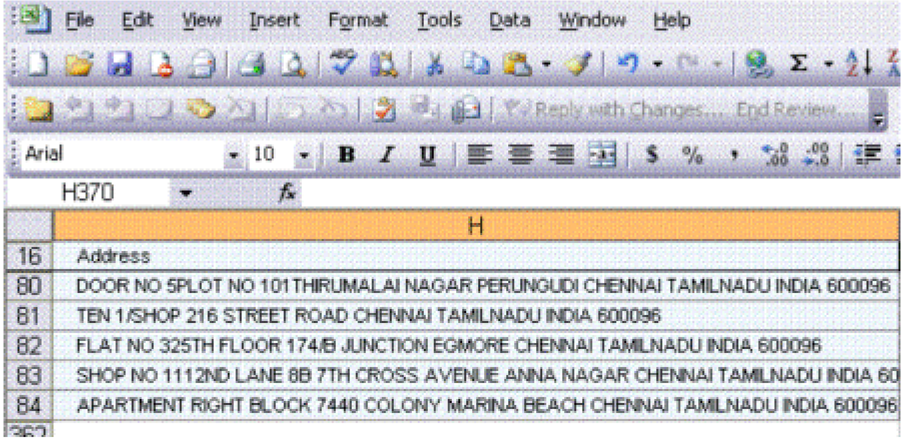
- Parameters needed to be changed in the Main Concatenation Program
  - findHeading – The Header of the column from which the concatenation needs to be started.
  - noOfColumns – Number of Columns which needs to be concatenated
  - startDataRange – This is mostly Constant at 16 , if the TABLE ANALYSIS GRID is placed at any another position this value needs to be changed
  - sheet – Name of the sheets seperated by commas
  - noOfSheets – Total number of sheets



```
Sub entryVariable(findHeading As String, startDataRange As Integer, noOfColumns As Integer, sheet As String, noOfSheet:

    findHeading = "Work Completed Indicator" 'Enter the Heading

    noOfColumns = "2" 'Enter number of Columns to be concatenated

    startDataRange = "16" 'Enter Starting row of records

    sheet = "DP3" 'Enter Sheet Names

    noOfSheets = "3" 'Enter Number of Sheets

End Sub
```

Note : In Case of multiple Sheets the existing table analysis grid must be deleted and new ones must be created thereby making the technical name of the Analysis GRID to be constant as GRID_1 , GRID_2 , and GRID_3 etc….. All other BEx functionalities remain same like filters and drilldown and drill across.

After concatenation:



	H
16	Address
80	DOOR NO 5PLOT NO 101THIRUMALAI NAGAR PERUNGUDI CHENNAI TAMILNADU INDIA 600096
81	TEN 1/SHOP 216 STREET ROAD CHENNAI TAMILNADU INDIA 600096
82	FLAT NO 325TH FLOOR 174/B JUNCTION EGMORE CHENNAI TAMILNADU INDIA 600096
83	SHOP NO 1112ND LANE 8B 7TH CROSS AVENUE ANNA NAGAR CHENNAI TAMILNADU INDIA 60
84	APARTMENT RIGHT BLOCK 7440 COLONY MARINA BEACH CHENNAI TAMILNADU INDIA 600096
362	

Related Content

Visual Basic for Applications

Learn all about Excel Macros (VBA)

Best Way - Excel VBA Tutorial

For more information, visit the Business Intelligence homepage.

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.