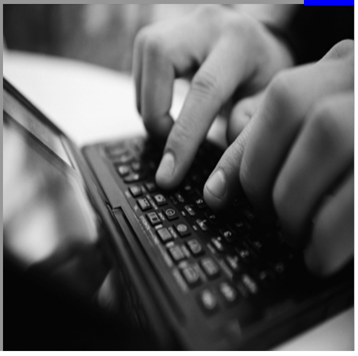


How-to Guide
SAP NetWeaver 7.0 (2004s)

How To... Build up a Simple Offline Planning Scenario in Bex Analyzer

Version 2.00 – December 2008

Applicable Releases:
SAP NetWeaver 7.0 (2004s)
SAP NetWeaver BI
BI Integrated Planning

THE BEST-RUN BUSINESSES RUN SAP

© Copyright 2007 SAP AG. 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 AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, and Informix are trademarks or registered trademarks of IBM Corporation in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

MaxDB is a trademark of MySQL AB, Sweden.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group 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.

These materials are provided “as is” without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. SAP shall not be liable for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials.
SAP does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third party web pages nor provide any warranty whatsoever relating to third party web pages.
SAP NetWeaver “How-to” Guides are intended to simplify the product implementation. While specific product features and procedures typically are explained in a practical business context, it is not implied that those features and procedures are the only approach in solving a specific business problem using SAP NetWeaver. Should you wish to receive additional information, clarification or support, please refer to SAP Consulting.
Any software coding and/or code lines / strings (“Code”) included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.

1 Scenario

In many companies users want to be able to have a simple offline planning solution. They want to open their normal Bex Analyzer Workbooks, save them locally on the disc and want to enter new data offline. Later – or maybe the next day - they want to log on to the system again by starting the Bex Analyzer, open the locally saved workbook, and save the changed data to the system. It is crucial that there is no separate offline solution but that the same workbooks as in online planning are used.

2 Introduction

This paper will show how to use some macros to enable a solution as described above. It is not the aim to build up a full blown offline planning application. Such an application would also require the offline execution of planning functions and a check out or locking mechanism for the data: if one user takes some data offline then no other user should be able to change them. The solution that we build up here will not have any such locking mechanism – it will be 'last one wins'. Nevertheless it is possible to extend the solution by using data slices to enable such a locking mechanism.

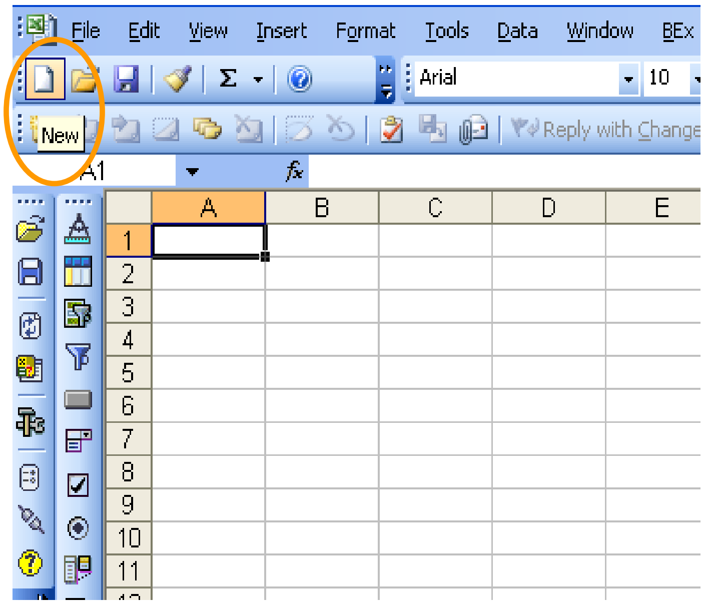
We will provide some macro coding that will be used instead of the standard refresh and save methods and enables the user to use the data that has been entered offline previously. The macro coding is generic and not depending on the number of analysis items or their names.

The coding also has some check functionality that ensures that during the time offline the query or the structure of the data has not been changed. Before writing back the changed data to the buffer the system checks whether the rows and the column headings in the offline data and the data on the server are still the same. If a row is missing or a column has been moved the system will stop the write back. As a result of this check functionality the solution does not offer the possibility to enter new lines in the offline mode.

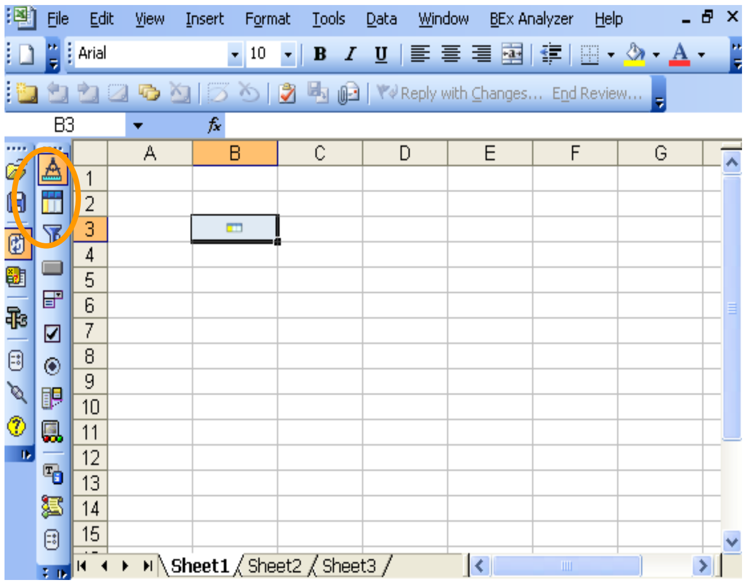
3 The Step By Step Solution

3.1 Create the Offline Workbook

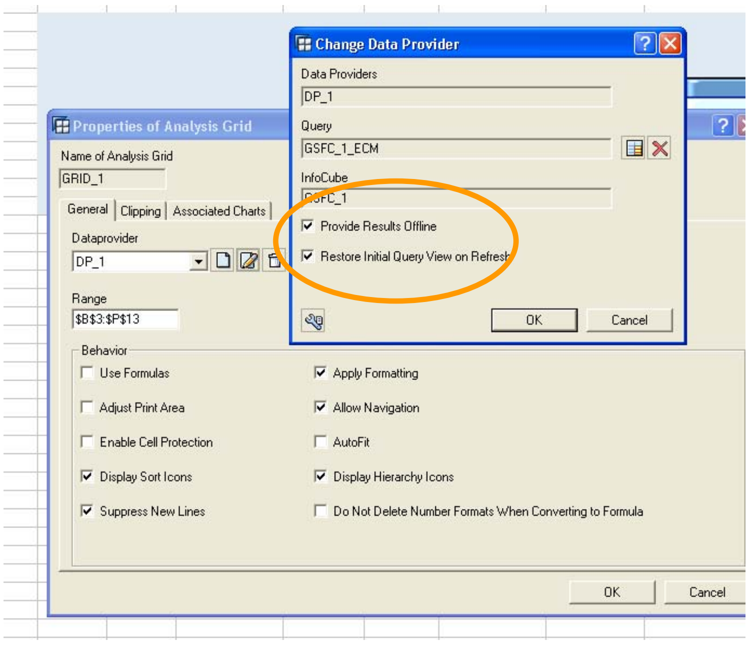
1. Create a new workbook: Open the Bex Analyzer and create a new (empty) workbook



2. Create an Analysis Item: In your workbook create an Analysis item using an input enabled query. [If you want to can add additional analysis item with other queries.]



3. Create a Data Provider for your Analysis Item: Create a Data Provider using an input enabled query. Make sure you set the flag 'Provide Result Offline'. For performance reasons you should also set the flag 'Restore Initial Query View on Refresh'.



Change Data Provider

Data Providers: DP_1

Query: GSFC_1_ECM

InfoCube: GSFC_1

- [✓] Provide Results Offline
- [✓] Restore Initial Query View on Refresh

OK | Cancel

Properties of Analysis Grid

Name of Analysis Grid: GRID_1

General | Clipping | Associated Charts

Dataprovider: DP_1

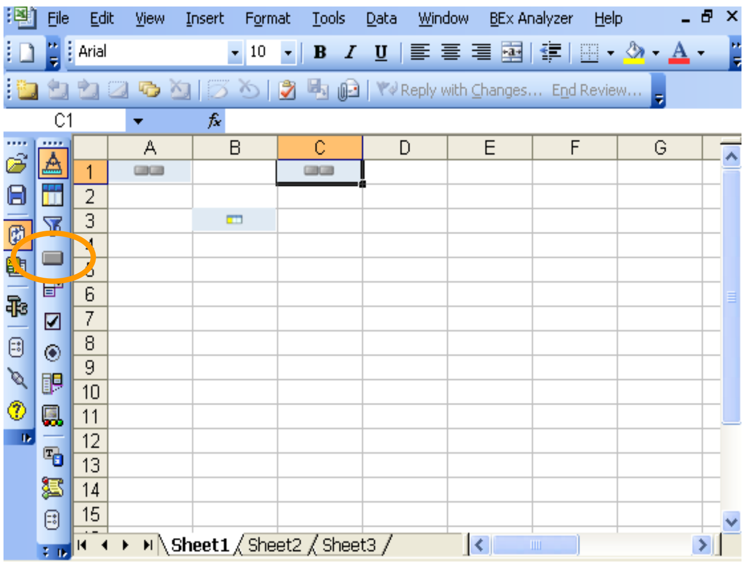
Range: $B$3:$P$13

Behavior

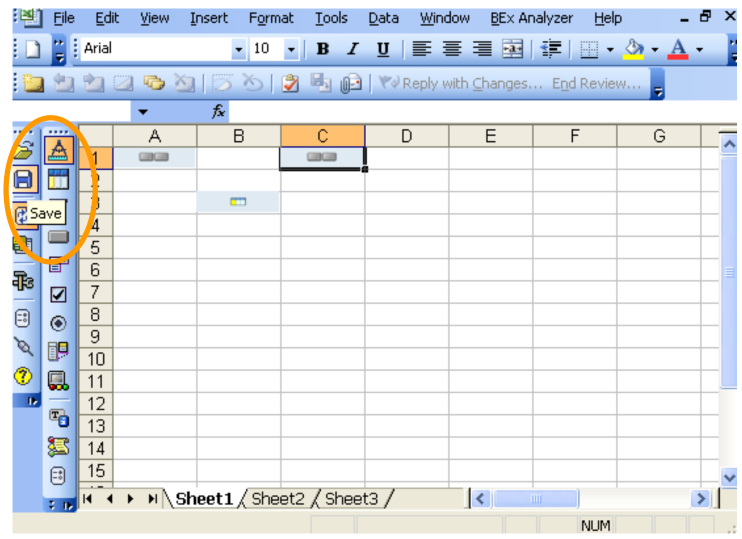
- [ ] Use Formulas
- [✓] Apply Formatting
- [ ] Adjust Print Area
- [✓] Allow Navigation
- [ ] Enable Cell Protection
- [ ] AutoFit
- [✓] Display Sort Icons
- [✓] Display Hierarchy Icons
- [✓] Suppress New Lines
- [ ] Do Not Delete Number Formats When Converting to Formula

OK | Cancel

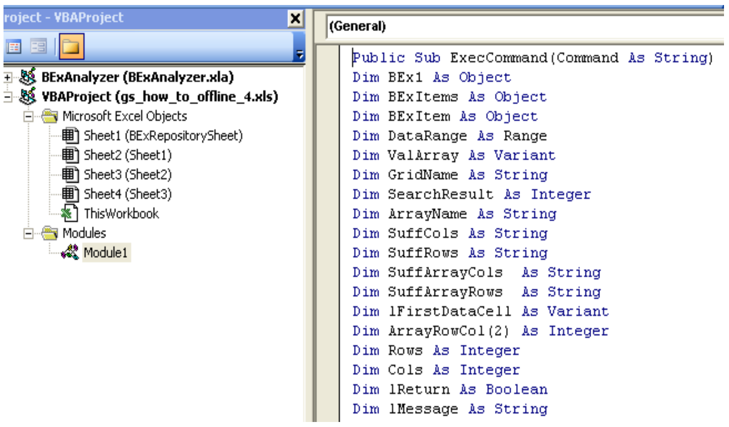
4. Create a refresh and a save button: Create a button group in your workbook. Use the command wizard to create a refresh button (transfer values) and a save button (planning specific commands).



5. Save your workbook: After the save for each button a macro is created. Instead of using step 3 and 4 you could create your own buttons using pure Excel tools. The disadvantage of this approach is that the buttons do not have the standard BI look.

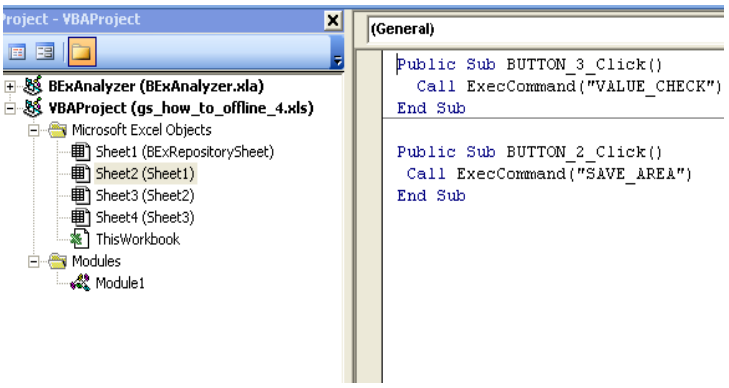


6. Add the macro coding from the appendix: Go to the Visual Basic Editor and create a new coding module. Copy the coding from the appendix into the new module.



```
Public Sub ExecCommand(Command As String)
Dim BEx1 As Object
Dim BExItems As Object
Dim BExItem As Object
Dim DataRange As Range
Dim ValArray As Variant
Dim GridName As String
Dim SearchResult As Integer
Dim ArrayName As String
Dim SuffCols As String
Dim SuffRows As String
Dim SuffArrayCols  As String
Dim SuffArrayRows  As String
Dim lFirstDataCell As Variant
Dim ArrayRowCol(2)  As Integer
Dim Rows As Integer
Dim Cols As Integer
Dim lReturn As Boolean
Dim lMessage As String
```

7. Exchange the coding of the buttons: In the VBA double click the sheet where you have created your buttons. You will find the macros generated for the buttons there. Depending on the technical name of the button (e.g. button 3) the macro is called 'BUTTON_3_Click()'. For the save button exchange the coding by: 'Call ExecCommand("SAVE_AREA")'.

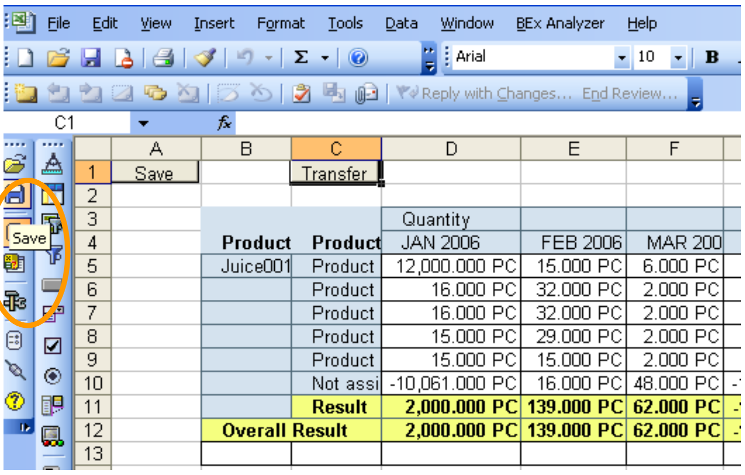


```
Public Sub BUTTON_3_Click()
  Call ExecCommand("VALUE_CHECK")
End Sub

Public Sub BUTTON_2_Click()
 Call ExecCommand("SAVE_AREA")
End Sub
```

8. Search the macro for the refresh button and replace the VBA coding by: Call ExecCommand("VALUE_CHECK").

9. Save your workbook again.



	A	B	C	D	E	F
1	Save		Transfer			
2						
3				Quantity		
4		Product	Product	JAN 2006	FEB 2006	MAR 200
5		Juice001	Product	12,000.000 PC	15.000 PC	6.000 PC
6			Product	16.000 PC	32.000 PC	2.000 PC
7			Product	16.000 PC	32.000 PC	2.000 PC
8			Product	15.000 PC	29.000 PC	2.000 PC
9			Product	15.000 PC	15.000 PC	2.000 PC
10			Not assi	-10,061.000 PC	16.000 PC	48.000 PC
11			**Result**	**2,000.000 PC**	**139.000 PC**	**62.000 PC**
12		**Overall Result**		**2,000.000 PC**	**139.000 PC**	**62.000 PC**
13						

Test the application. Open you workbook and enter some data. Check that the refresh and the save button work correctly in the standard online mode.

Now you can test the offline capability. Open the workbook, save a copy to your disc and close the workbook and the Bex Analyzer. Open you local copy and enter some data (remember that no new lines are supported). Save and close the workbook. Now start the Bex Analyzer again and open you local copy with the changed data. Press refresh or save. The data that has been entered offline will be written correctly to the delta buffer and data base. [Instead of opening and closing the workbook and the Bex Analyzer you can also disconnect the Bex Analyzer, enter the offline data and reconnect again].

As the macro will detect the Analysis items generically you can add as further analysis items without changing the solution. The Analysis Item can be placed in any of the worksheets.

The above solution uses an internal buffering mechanism to correctly transfer the data from the planning enabled grids to the server. If a data provider is not planning enabled no buffering is done (as it is unnecessary) and a normal refresh is done.

4 Appendix

In this section we include the coding that we have used above.

```
Public Sub ExecCommand(Command As String)
Dim BEx1 As Object
Dim BExItems As Object
Dim BExItem As Object
Dim DataRange As Range
Dim ValArray As Variant
Dim GridName As String
Dim SearchResult As Integer
Dim ArrayName As String
Dim SuffCols As String
Dim SuffRows As String
Dim SuffArrayCols  As String
Dim SuffArrayRows  As String
Dim lFirstDataCell As Variant
Dim ArrayRowCol(2) As Integer
Dim Rows As Integer
Dim Cols As Integer
Dim lReturn As Boolean
Dim lMessage As String
Dim lActiveSheet As String
Dim lRefreshNecessary As Boolean
Dim bexparam As Object
Dim lInputSwitchFound As Boolean
Dim lSwitchedGrids As String

SuffCols = "ColVals"
SuffRows = "RowVals"
SuffArrayCols = "ColArray"
SuffArrayRows = "RowArray"

lActiveSheet = Application.ActiveSheet.Name

Application.ScreenUpdating = False
Set BEx1 = Application.Run("BExAnalyzer.xla!GetBEx")
On Error GoTo ConnectionError

Set BExItems = BEx1.Items

'Save data to internal tables
Set LTabs = CreateObject("Scripting.Dictionary")

For Each BExItem In BExItems

'Check that we have a grid!!
  SearchResult = InStr(1, BExItem.Name, "GRID")
  If SearchResult <> 0 Then

'we save all data from input enabled grids into our scripting dictionary
```

```
  If BExItem.DataProvider.Request.Properties.AllowInput = True Then

  lRefreshNecessary = True

' set the correct worksheet
  Application.Worksheets(BExItem.Worksheet.Name).Activate

' data
  Set DataRange = getDataRange(BExItem)
  GridName = BExItem.Name

  With DataRange
    ValArray = Range(Cells(.Row, .Column), Cells(.Row + .Rows.Count - 1, .Column +
.Columns.Count - 1)).Value
  End With
  LTabs.Add GridName, ValArray

  lFirstDataCell = BExItem.DataProvider.Result.Grid.firstdatacell

  With BExItem.Range
' Rows
    ArrayName = GridName + SuffRows
    ValArray = Range(Cells(.Row, .Column), Cells(.Row + .Rows.Count - 1, .Column +
lFirstDataCell.x - 1)).Value
    LTabs.Add ArrayName, ValArray

    ArrayName = GridName + SuffArrayRows
    ArrayRowCol(0) = .Rows.Count
    ArrayRowCol(1) = lFirstDataCell.x
    LTabs.Add ArrayName, ArrayRowCol

' Columns
    ArrayName = GridName + SuffCols
    ValArray = Range(Cells(.Row, .Column), Cells(.Row + lFirstDataCell.y - 1, .Column +
.Columns.Count - 1)).Value
    LTabs.Add ArrayName, ValArray

    ArrayName = GridName + SuffArrayCols
    ArrayRowCol(0) = lFirstDataCell.y
    ArrayRowCol(1) = .Columns.Count
    LTabs.Add ArrayName, ArrayRowCol

  End With

  End If
  End If
  Next

' we check whether anything has to be done at all - no input item found - ne refresh: do a
transmit and end
  If lRefreshNecessary = False Then
' do a normal transfer and end the processing
    Set bexparam = CreateObject("com.sap.bi.et.analyzer.api.BExParameter")
    bexparam.Add "WORKBOOK_NAME", BEx1.Name
    bexparam.Add "CMD", Command
    Call BEx1.Process(bexparam)
    Call BEx1.FrontendUpdate
    Call BEx1.Render

    Application.Worksheets(lActiveSheet).Activate
    Application.ScreenUpdating = True
    Exit Sub
  End If


' we have at least one offline input item; go on with first refresh
' refresh
' BEx1.Refresh (True)
  BEx1.Synchronize = True
'Rendering is necessary for delta check on cell level.
'If entire ranges are pasted (which should not be done when distributions are used or cell
protection)
' then this call is not necessary
  Call BEx1.Render

' return the data
' reset the check flag for update necessary
lRefreshNecessary = False

' before transferring the data we have to check that the lead columns and the
' rows have not changed
  For Each BExItem In BExItems
  SearchResult = InStr(1, BExItem.Name, "GRID")
  If SearchResult <> 0 Then

' we might have a grid that is now input off but was input on when workbook was taken
offline
' check in the scripting dictionary - we simply use the columns for checking
  ArrayName = BExItem.Name + SuffCols
  If IsEmpty(LTabs.Item(ArrayName)) = False Then
' the current grid was input enabled when we started the offline - is it still input enabled?
' check whether the new array is input on


  If BExItem.DataProvider.Request.Properties.AllowInput = False Then
' the new array is input off - there has been a switch (e.g. locking problem)
' set the names for the message
  If lInputSwitchFound = False Then
' first switched grid
    lSwitchedGrids = BExItem.Name
  Else
    lSwitchedGrids = lSwitchedGrids + ", " + BExItem.Name
  End If
  lInputSwitchFound = True

  Else

' we know that the grid is still input on and we restore the data
' set the correct worksheet
```

```
  Application.Worksheets(BExItem.Worksheet.Name).Activate

  Set DataRange = getDataRange(BExItem)
  GridName = BExItem.Name

  lFirstDataCell = BExItem.DataProvider.Result.Grid.firstdatacell

  With BExItem.Range
' Rows
' compare the arrays
    ArrayName = GridName + SuffArrayRows
    Rows = LTabs.Item(ArrayName)(0)
    Cols = LTabs.Item(ArrayName)(1)

    ArrayName = GridName + SuffRows

  lReturn = compareArrays(Range(Cells(.Row, .Column), Cells(.Row + .Rows.Count - 1,
.Column + lFirstDataCell.x - 1)).Value, .Rows.Count, lFirstDataCell.x,
LTabs.Item(ArrayName), Rows, Cols)
    If lReturn = False Then
      lMessage = "The row structure or content of " + GridName + " has changed. An
update is not possible"
      MsgBox lMessage
      'MsgBox "The row structure or content has changed. An update is not possible"
      Exit Sub
    End If

' Columns
    ArrayName = GridName + SuffArrayCols
    Rows = LTabs.Item(ArrayName)(0)
    Cols = LTabs.Item(ArrayName)(1)

    ArrayName = GridName + SuffCols

  lReturn = compareArrays(Range(Cells(.Row, .Column), Cells(.Row + lFirstDataCell.y -
1, .Column + .Columns.Count - 1)).Value, lFirstDataCell.y, .Columns.Count,
LTabs.Item(ArrayName), Rows, Cols)
    If lReturn = False Then
      lMessage = "The column structure or content of " + GridName + " has changed. An
update is not possible"
      MsgBox lMessage
      'MsgBox "The column structure or content has changed. An update is not possible"
      Exit Sub
    End If
  End With


'Update the data
Dim CompArray As Variant
ReDim CompArray(DataRange.Rows.Count, DataRange.Columns.Count)

  With DataRange
   CompArray = Range(Cells(.Row, .Column), Cells(.Row + .Rows.Count - 1, .Column +
.Columns.Count - 1)).Value
   For i = 0 To .Rows.Count - 1
```

```
    For j = 0 To .Columns.Count - 1
      If CompArray(i + 1, j + 1) <> LTabs.Item(GridName)(i + 1, j + 1) Then
        If Range(Cells(.Row + i, .Column + j), Cells(.Row + i, .Column + j)).Locked = False
Then
' something has changed - we need to refresh
           lRefreshNecessary = True
           Range(Cells(.Row + i, .Column + j), Cells(.Row + i, .Column + j)).Value =
LTabs.Item(GridName)(i + 1, j + 1)
        End If
      End If
     Next j
   Next i

'Original version (problems with distribution and potentially with protection):
'Update the data
' With DataRange
'   Range(Cells(.Row, .Column), Cells(.Row + .Rows.Count - 1, .Column +
.Columns.Count - 1)).Value = LTabs.Item(GridName)

  End With

  End If
  End If
  End If
  Next

' transfer if necessary
  If lRefreshNecessary = True Then
    Set bexparam = CreateObject("com.sap.bi.et.analyzer.api.BExParameter")
    bexparam.Add "WORKBOOK_NAME", BEx1.Name
    bexparam.Add "CMD", Command

    Call BEx1.Process(bexparam)
    Call BEx1.FrontendUpdate
    Call BEx1.Render
  End If

  Application.Worksheets(lActiveSheet).Activate
  Application.ScreenUpdating = True

' if the input has switched before then we inform the user that his data is lost
  If lInputSwitchFound = True Then
    lMessage = "The input readiness of the grid(s) " + lSwitchedGrids + " has changed. An
update is not possible. Please check the messages for information about locks and data
slices."
      MsgBox lMessage
  End If


  Exit Sub

ConnectionError:
 MsgBox "Error. Please log on and try again."

End Sub
```

```
Private Function getDataRange(GridItem As Object) As Range
  Dim lFirstDataCell As Variant
  lFirstDataCell = GridItem.DataProvider.Result.Grid.firstdatacell
  With GridItem.Range
    Set getDataRange = Range(Cells(.Row + lFirstDataCell.y, .Column + lFirstDataCell.x),
Cells(.Row + .Rows.Count - 1, .Column + .Columns.Count - 1))
  End With
End Function


Private Function compareArrays(iArray1 As Variant, iRows1 As Integer, iCols1 As
Integer, iArray2 As Variant, iRows2 As Integer, iCols2 As Integer) As Boolean

 compareArrays = True

Dim i As Integer
Dim j As Integer

' check the size
If iRows1 <> iRows2 Then
 compareArrays = False
 Exit Function
End If

If iCols1 <> iCols2 Then
 compareArrays = False
 Exit Function
End If

' check the values
If iRows1 = 1 And iCols1 = 1 Then
  If iArray1 <> iArray2 Then
     compareArrays = False
     Exit Function
  End If
Else
  For i = 1 To iRows1
    For j = 1 To iCols1
     If iArray1(i, j) <> iArray2(i, j) Then
       compareArrays = False
       Exit Function
     End If
    Next
  Next
End If

End Function
```

www.sdn.sap.com/irj/sdn/howtoguides

THE BEST-RUN BUSINESSES RUN SAP