

Automated Dynamic Data Exchange (DDE) Replacement Solution for SAS® GRID

Ajay Gupta, PPD, Morrisville, USA

ABSTRACT

In the pharmaceutical and contract research organization (CRO) industries, Microsoft Excel is widely used to create mapping specs and in reporting. SAS® enables you to export data into an Excel spreadsheet and do Excel formatting in many ways. But, Dynamic Data Exchange (DDE) is the only technique that provides total control over the Excel output. DDE uses a client/server relationship to enable a client application to request information from a server application. SAS is always the client. In this role, SAS requests data from server applications, sends data to server applications, or sends commands to server applications. Unfortunately, DDE is not supported on SAS® Grid Computing. This paper explores a replacement solution for DDE on a SAS grid by using a SAS® Stored Process, Microsoft Visual Basic for Applications (VBA), and SAS® Add-in for Microsoft Office. The paper also explores the automation process and extends the solution to format Microsoft Word documents.

INTRODUCTION

Dynamic Data Exchange (DDE) is a Microsoft protocol for exchanging information between applications on the same Windows machine. For SAS users, it allows programs to interact with Excel workbooks and other Microsoft Office documents; however, it is gradually becoming obsolete as it is incompatible with modern grid computing. For many years DDE had served as a critical tool in our organization for generating high quality and consistent Excel tables on a large scale; however, when our organization made the decision to move from individual SAS PC licenses to Windows SAS grid computing, we needed an alternative approach. While SAS now provides several other ways to interact with Excel, which we also use, it does not allow writing data to specific ranges in formatted Excel templates, deletion of Excel tabs, and color formatting. Many of the output produced for clients incorporate a variety of disparate pieces of information from multiple SAS procedures, and DDE provided a convenient method of sequentially adding output to an Excel table.

This paper will initially provide some workaround solutions to perform DDE like functionality. Later, we explore a replacement solution for DDE on a SAS grid by using a SAS® Stored Process, Microsoft Visual Basic for Applications (VBA), and SAS® Add-in for Microsoft Office. The paper also explores the automation process and extends the solution to format Microsoft Word documents.

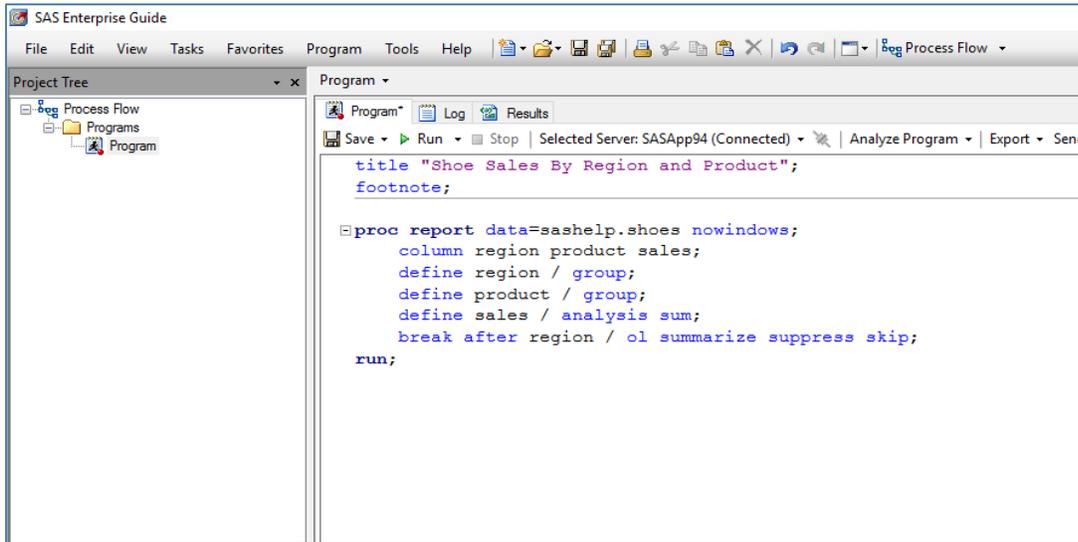
TWO-STEP DDE REPLACEMENT SOLUTION

The easiest way to get the DDE type functionality is to follow a two-step process. This process later becomes the basic building block for full automated solution.

1. Execute SAS program or stored process separately to create the output in Excel. Here we will explore the stored process which is building block for automated solution.
2. Later, execute the VBA macro manually through Excel. This method will give the user full control over the Excel output.

Now, we will explore the two-step process which is the building block for the automated replacement solution.

Sample SAS Code: Consider the following sample SAS program to get the shoe sales by Region and product. We will use this sample code for all the solutions given below.

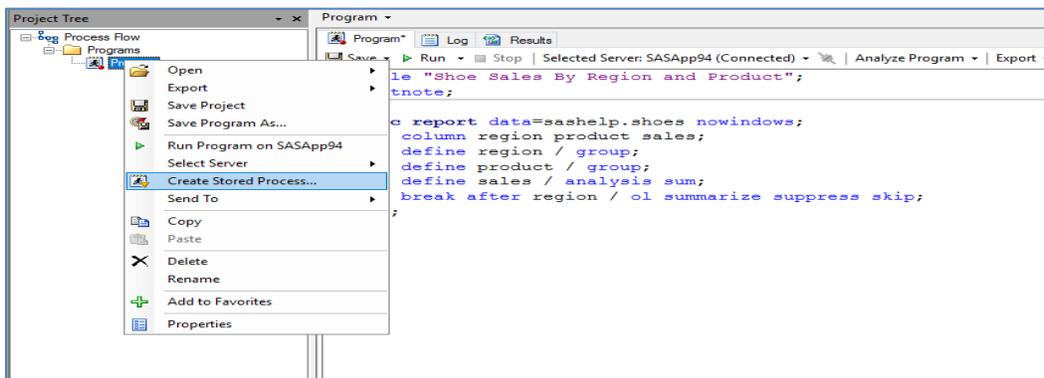


Display 1. Sample SAS program

Create SAS Stored Process:

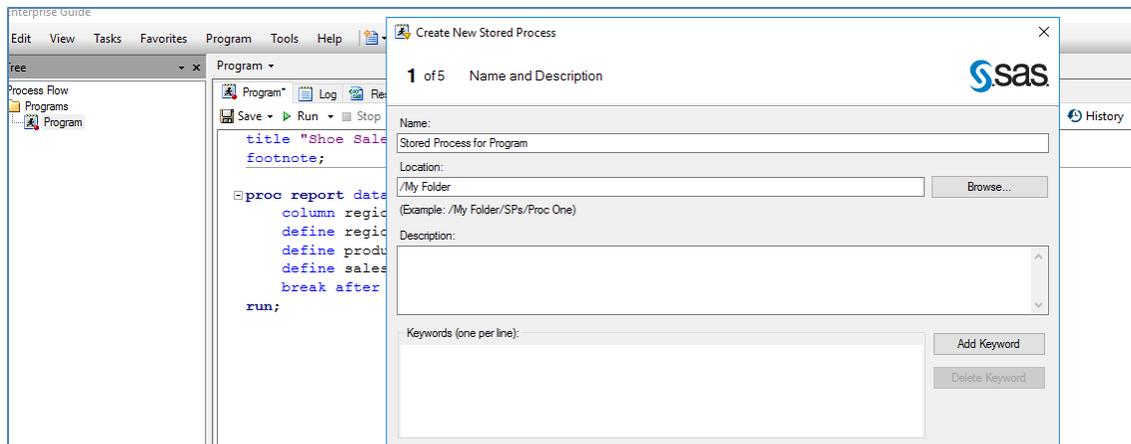
Users can easily create a SAS stored process using SAS® Enterprise Guide. There are multiple papers on the SAS website (also added a paper link in references) which will walkthrough users on how to create a stored process using SAS® Enterprise Guide. So, the author is only planning to cover a few initial steps.

Step 1: Open the program in Enterprise Guide and then right click on the program and select Create Stored Process.



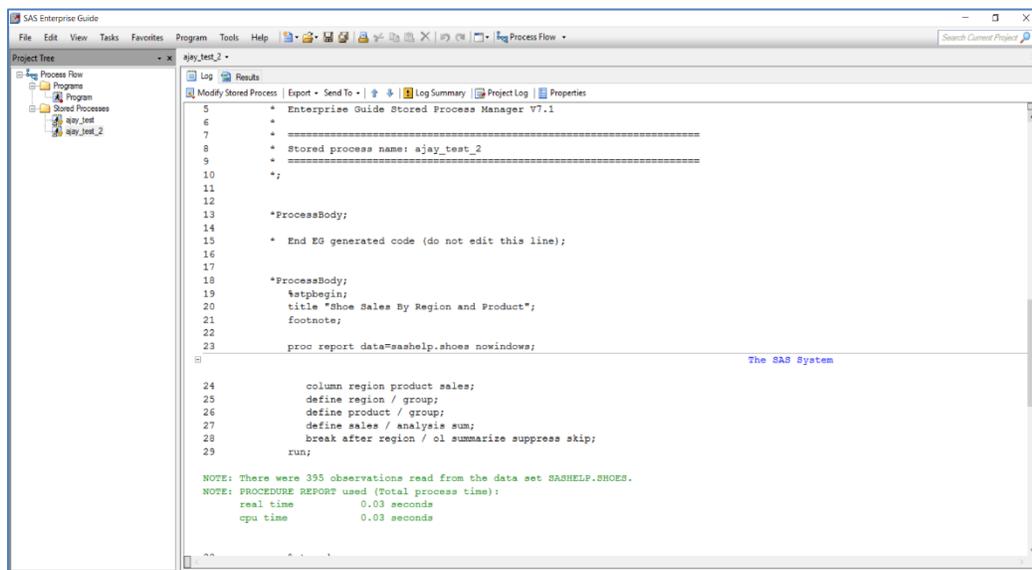
Display 2. Create Stored Process in SAS® Enterprise Guide

Step 2: Chose the program name and Location (see display 3) and complete the next 5 steps on the screen to create the stored process, e.g. Ajay_Test_2.



Display 3. Steps to create the stored process in SAS® Enterprise Guide

Step 3: The new stored process, e.g. Ajay_Test_2, containing the sample code can be accessed through stored processes under process flow. Execute the stored process to make sure it is running without error. The log will have the stored process code. See display 4 below for sample code. Normally, the stored process code is nothing, but the original SAS program code encapsulated between two macros %STPBEGIN and %STPEND.



Display 4. SAS stored process log with sample code.

RECORD VBA MACRO IN EXCEL FOR FORMATTING:

For many, perhaps most SAS developers the easiest and fastest way to write VBA code is to first go into a Microsoft application (in this paper Microsoft Excel and Microsoft Word are used as the application), turn on the macro recorder, perform the steps and functions desired, and then terminate the macro recorder. The end result is a stored VBA program (macro) that can be executed at a later date. This makes it possible for a SAS developer to automate tasks in the Microsoft application, and therefore vastly improve the functionality of an integrated system that takes advantage of the relative strengths of the SAS System and the Microsoft application.

Steps to record VBA Macro for page formatting are as follows:

1. Open Excel.

2. Select Tools, Macro, and Record New Macro from the pull-down menus.
3. Enter the name of macro as 'Format_VBA'.
4. Select Ok.
5. Select the stop recording button.
6. Select Tools, Macro, and Macros from the pull-down menus and select 'Format_VBA' in edit mode you will see the Visual Basic Editor.
7. Copy the Format_VBA macro code given below and save. Further, close the Excel. User can further modify the code as per there requirements. This code will do some color formatting and freeze the pan on row 4 to easily scroll through results.

Similar process can be followed to create other macros given in the paper. VBA code for Format_VBA macro is given below in Display 5:

```

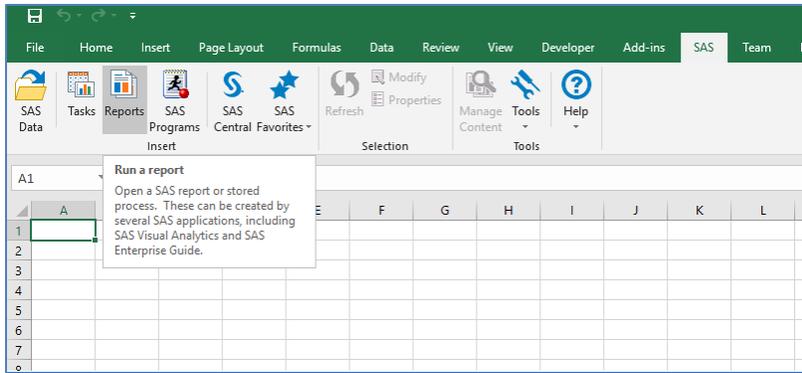
Sub Format_VBA()
'
' Format_VBA Macro
'
'
Range("A1:C1").Select
Selection.Font.Bold = True
With Selection.Interior
.Pattern = xlSolid
.PatternColorIndex = xlAutomatic
.Color = 255
.TintAndShade = 0
.PatternTintAndShade = 0
End With
Range("A3:C3").Select
Selection.Font.Bold = True
With Selection.Interior
.Pattern = xlSolid
.PatternColorIndex = xlAutomatic
.Color = 65535
.TintAndShade = 0
.PatternTintAndShade = 0
End With
Rows("4:4").Select
ActiveWindow.FreezePanes = True
End Sub

```

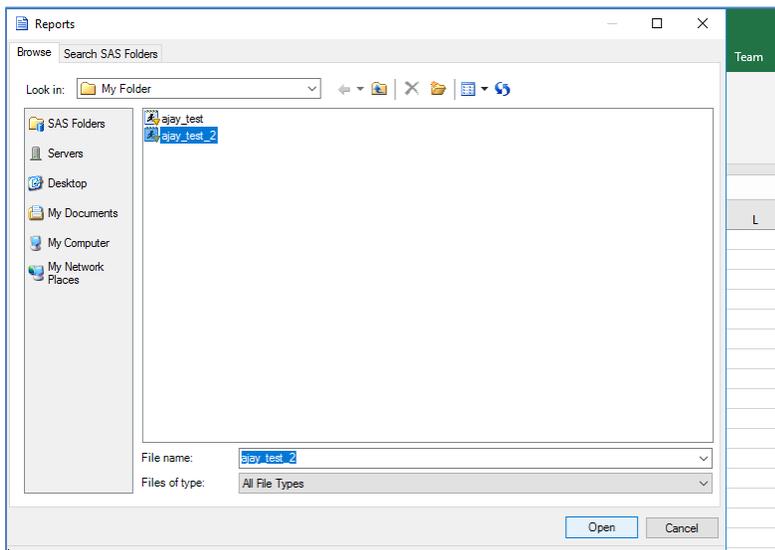
Display 5. Format_VBA macro code to do excel formatting.

EXECUTE STORED PROCESS AND VBA MACRO SEPERATELY IN MICROSOFT EXCEL ADD-IN

Step 1: Open Microsoft Excel and go to SAS Add-in tab. Later, click Reports and explore the SAS Stored Process, e.g. Ajay_Test_2 (see display 6 and 7). Once the stored process is located and selected then click open. This will invoke the stored process in Excel.

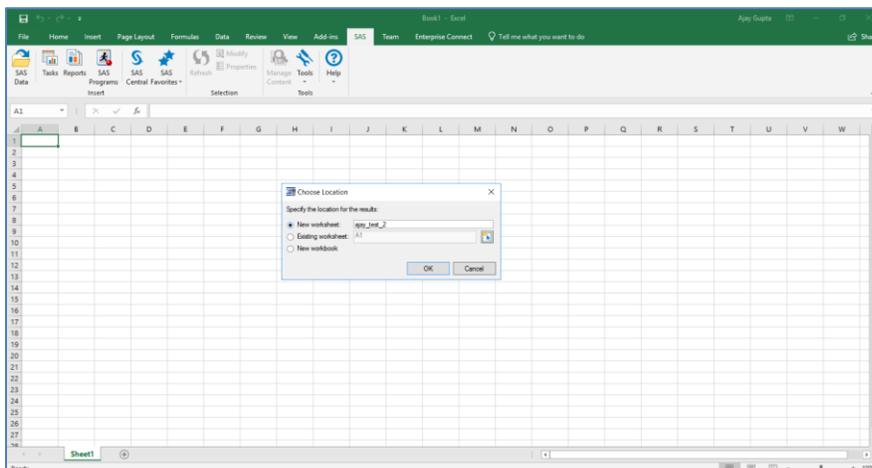


Display 6. Step to open SAS Stored Process in Microsoft add-in

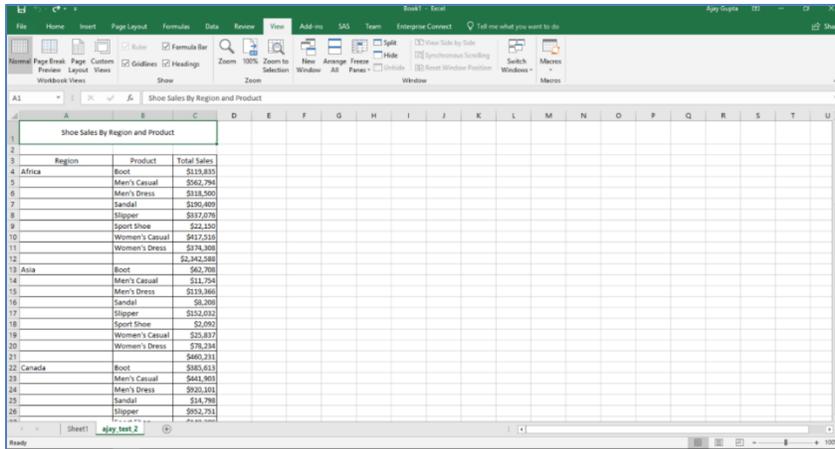


Display 7. Step to explore SAS Stored Process in Microsoft add-in

Step 2: Once the stored process is executed then a window will pop open asking for the location of results (see display 8). Select the location of results and click 'Ok'. Later, sales by region and product will be generated in Excel (see display 9).

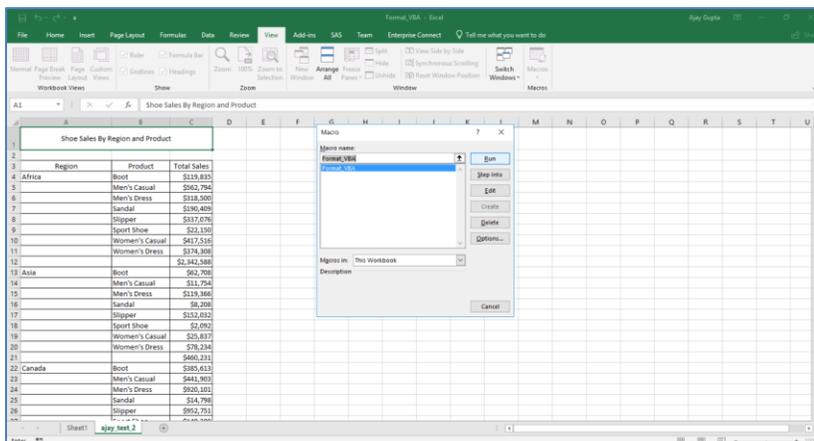


Display 8. Window for the location of results

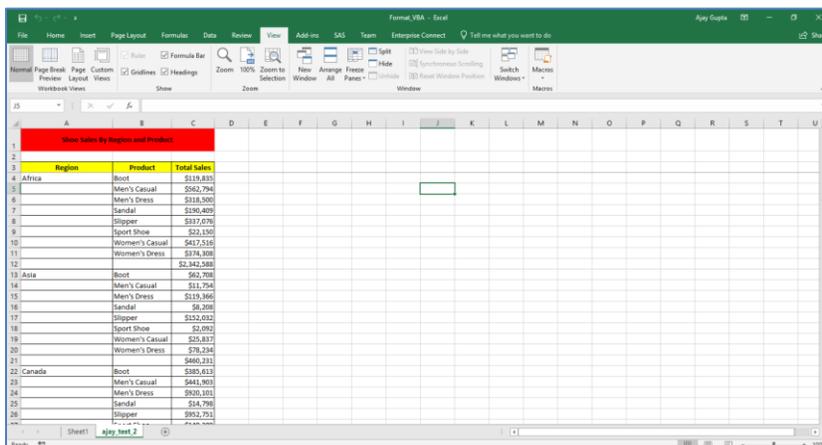


Display 9. Results in Excel after the execution

Step 3: Select Tools, Macro, and View Macros from the pull-down menus. Later, select VBA macro Format_VBA and click run (see display 10). After execution, the VBA macro will apply formatting to the Excel tab as shown in display 11. The user can later save the output programmatically through VBA using the save command.



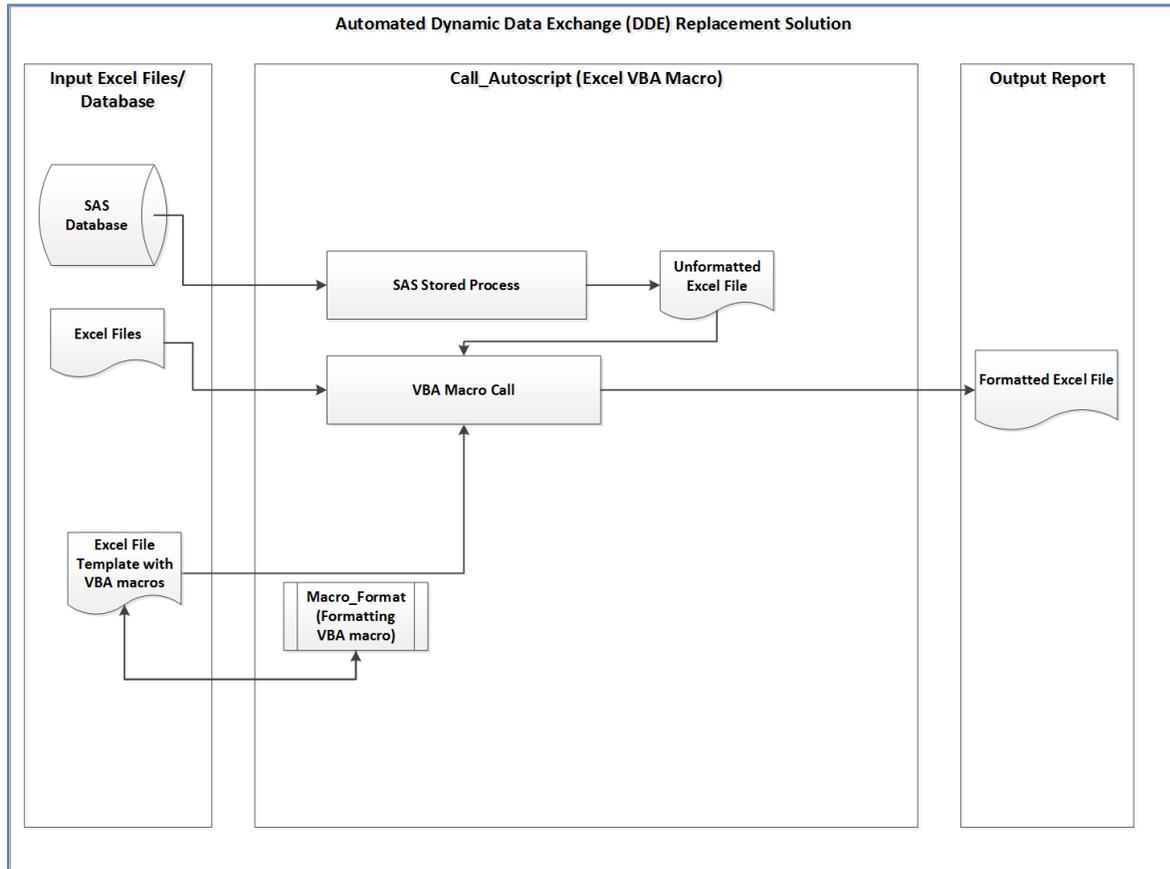
Display 10. Execute Format_VBA macro.



Display 11. Final output with color formatting

AUTOMATED DYNAMIC DATA EXCHANGE (DDE) REPLACEMENT SOLUTION

See below the flow diagram for the process flow and components involved in automating the DDE replacement solution. Basically, it is a further extension of the two-step process where all the steps are automatically invoked through a VBA macro within Microsoft SAS add-in. Now, we will explore the automated DDE replacement solution step by step.

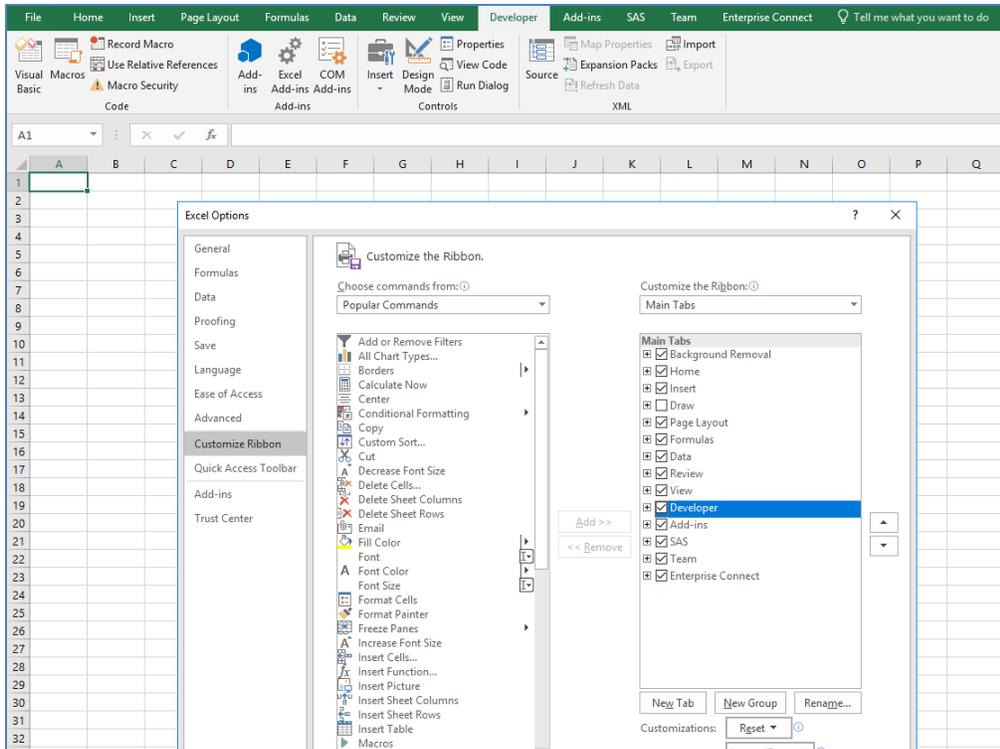


Display 12. Automated DDE replacement solution flow diagram

ADD SAS ADD-IN OBJECT IN MICROSOFT EXCEL

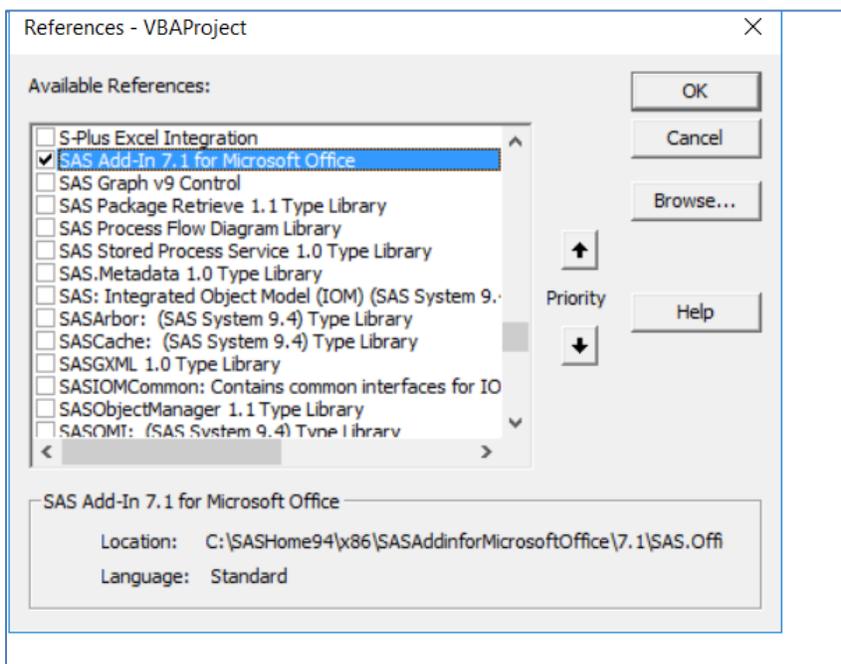
To invoke SAS stored process from VBA we need to add SAS Add-in object in Excel using the steps below. This will be helpful to invoke a SAS type object through VBA.

Step 1: You first need to follow these steps in order to set up Microsoft Excel to allow you to use the Visual Basic functionality. To turn on the **Developer** tab with Office 2010, select **File > Options > Customize Ribbon**. Make sure that **Developer** is selected on the **Main Tabs** list in the right selection box. See display 13 below for detail.



Display 13. Add developer tab in Excel to access VBA functionality.

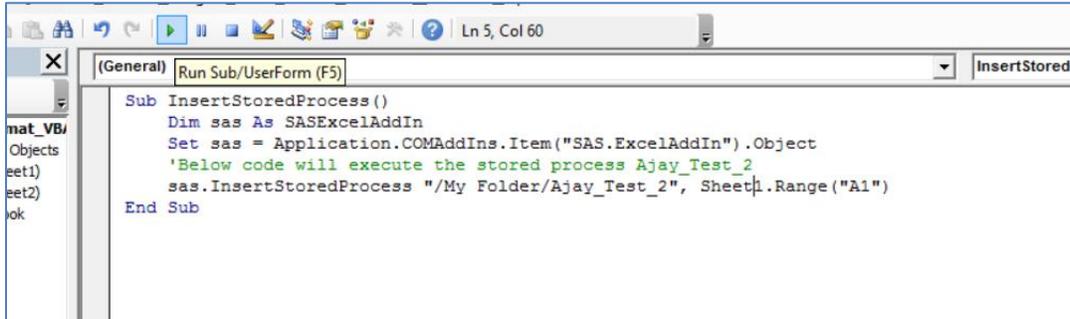
Step 2: Select the SAS Add-In Object Reference type library so that Microsoft Excel can find the SAS objects that are referred to by the Visual Basic code. Select **Developer > Visual Basic > Tools > References**. Scroll down the list and select the **SAS Add-In 7.1 for Microsoft Office** check box (whichever version of SAS Add-in is applicable). See display 14 for detail.



Display 14. Add SAS Add-in in Excel VBA functionality

INVOKE SAS STORED PROCESS THROUGH EXCEL USING VBA MACRO

Step 1: To invoke a SAS stored process from VBA, create the VBA macro InsertStoredProcess using the process like the one described above for VBA macro Format_VBA. See below display for InsertStoredProcess VBA macro code. Later, run the macro using the run button shown below.

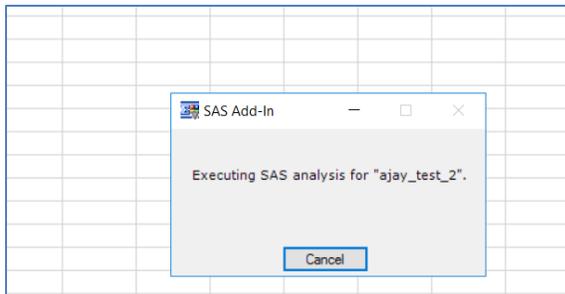


```

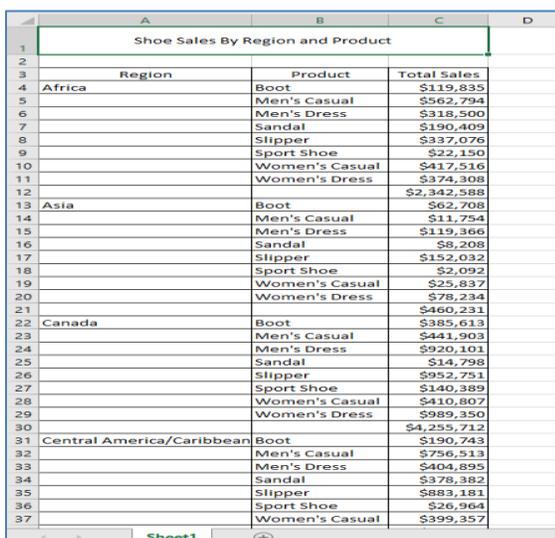
Sub InsertStoredProcess ()
    Dim sas As SASExcelAddIn
    Set sas = Application.COMAddIns.Item("SAS.ExcelAddIn").Object
    'Below code will execute the stored process Ajay_Test_2
    sas.InsertStoredProcess "/My Folder/Ajay_Test_2", Sheet1.Range("A1")
End Sub
    
```

Display 15. InsertStoredProcess VBA macro

Step 2: Once the InsertStoredProcess VBA macro is executed it will execute the stored process, e.g. Ajay_Test_2 (see display 16) and later output the results in sheet1 or a given location, e.g. see display 17.



Display 16. SAS stored process execution through VBA

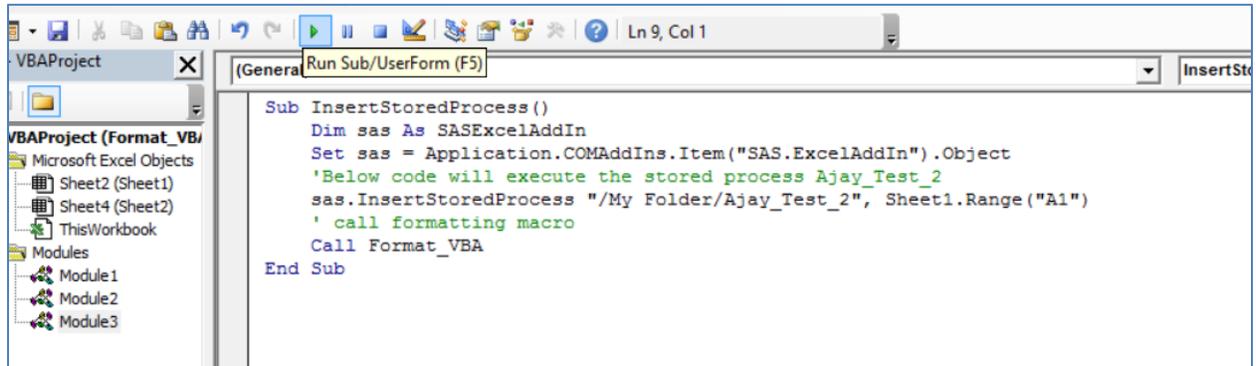


Shoe Sales By Region and Product		
Region	Product	Total Sales
Africa	Boot	\$119,835
	Men's Casual	\$562,794
	Men's Dress	\$318,500
	Sandal	\$190,409
	Slipper	\$337,076
	Sport Shoe	\$22,150
	Women's Casual	\$417,516
	Women's Dress	\$374,308
		\$2,342,588
Asia	Boot	\$62,708
	Men's Casual	\$11,754
	Men's Dress	\$119,366
	Sandal	\$8,208
	Slipper	\$152,032
	Sport Shoe	\$2,092
	Women's Casual	\$25,837
	Women's Dress	\$78,234
		\$460,231
Canada	Boot	\$385,613
	Men's Casual	\$441,903
	Men's Dress	\$920,101
	Sandal	\$14,798
	Slipper	\$952,751
	Sport Shoe	\$140,389
	Women's Casual	\$410,807
	Women's Dress	\$989,350
		\$4,255,712
Central America/Caribbean	Boot	\$190,743
	Men's Casual	\$756,513
	Men's Dress	\$404,895
	Sandal	\$378,382
	Slipper	\$883,181
	Sport Shoe	\$26,964
	Women's Casual	\$399,357

Display 17. Stored Process final output in Sheet1

AUTOMATE SAS STORED PROCESS AND VBA MACRO EXECUTION IN EXCEL

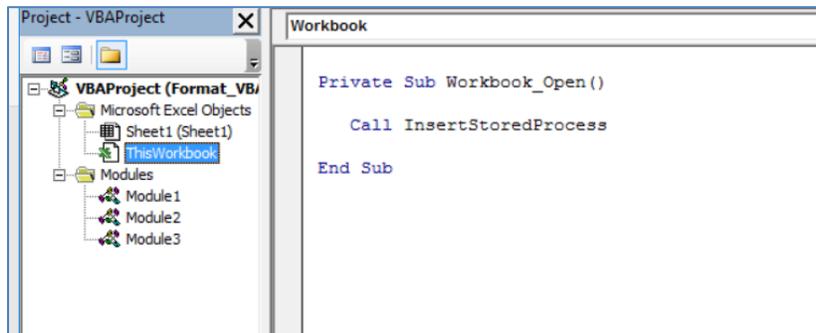
Step 1: To invoke a SAS stored process and VBA macro together, add a call for Format_VBA within InsertStoredProcess. See below display for the InsertStoredProcess VBA macro code.



Display 18. Stored Process and VBA macro call together

Step 2: To invoke a SAS stored process and VBA macro in InsertStoredProcess automatically follow the steps below as soon as the macro enabled file is opened.

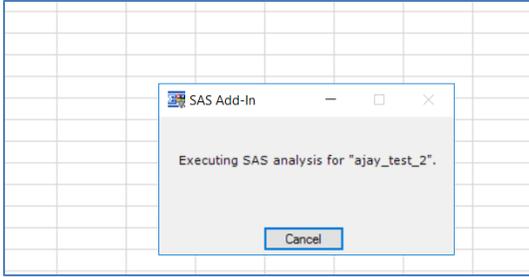
- Select Thisworkbook and later select Workbook from the drop-down menu on the right side.
- This will create a macro code template for Workbook_Open.
- Add a call for InsertStoredProcess. See below display.
- Save the file as a macro enabled file named DDE_AUTOSCRIPPT



Display 19. Automate the process Workbook_open macro

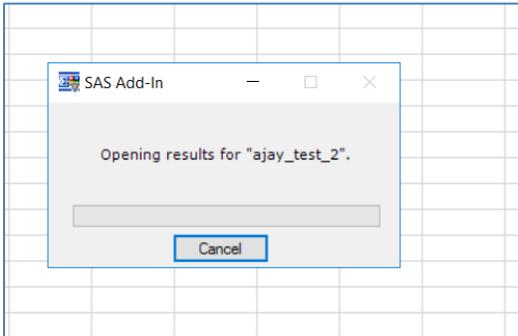
Step 3: Open the file DDE_AUTOSCRIPPT. The Workbook_open macro will automatically be executed and will further invoke the SAS Stored Process and Format_VBA macro, to create a formatted Excel output in a given location. The whole process can be described as follows.

1. The auto script will automatically execute the SAS stored process, e.g. Ajay_Test_2. See display below.



Display 20. Stored Process invoke through Auto script

- The stored process will create the unformatted output in Sheet 1. See display 21 below.



Display 21. Stored Process final unformatted output in Sheet1

- Later, Format_VBA will be executed to do the color formatting or DDE functionality and create a final formatted output. See display 22 below.

Shoe Sales By Region and Product		
Region	Product	Total Sales
Africa	Boot	\$119,835
	Men's Casual	\$562,794
	Men's Dress	\$318,500
	Sandal	\$190,409
	Slipper	\$337,076
	Sport Shoe	\$22,150
	Women's Casual	\$417,516
	Women's Dress	\$374,308

Display 22. Final formatted output after the execution of FORMAT_VBA macro.

ADDITIONAL ENHANCEMENT TO AUTOMATED DDE REPLACEMENT SOLUTION

There is a possibility that a user is not familiar with VBA programming and might be looking for a more flexible solution in which they can pass different values (on demand) to format

the Excel output. In that case the above solution can be enhanced as follows:

Step 1: Create a stored process which will output SASHELP.CLASS in worksheet class. Later, add code in the stored process to create tab VBA_Command in Excel file which will have the following columns: Sheet, Cell, and Color. This file can be expanded further to accommodate different commands, e.g. change font size, color, change case etc. See display 23 below for more detail.

	A	B	C	D	E
1	Sheet	cell	color		
2	CLASS	C2	3		
3	CLASS	D2	5		
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					

Display 23. Stored Process VBA output

Step 2: Open the macro Format_VBA using the developer tab and replace the code with the one below. This code can be further expanded as per requirements. Later, save the auto script file.

```

Sub Format_VBA ()
    Dim Excelsource As String
    Dim Excelcell As String
    Dim Excelcommand As String

    ActiveWorkbook.Worksheets("VBA_COMMAND").Select
        k = Cells(Rows.Count, "A").End(xlUp).Row

    For irow = 2 To k
        'Get value from cell

        Excelsource = ActiveWorkbook.Worksheets("VBA_COMMAND").Range("A" & irow)
        Excelcell = ActiveWorkbook.Worksheets("VBA_COMMAND").Range("B" & irow)
        Excelcommand = ActiveWorkbook.Worksheets("VBA_COMMAND").Range("C" & irow)

        'Apply Formatting

        Worksheets(Excelsource).Select
        Range(Excelcell).Select

            With Selection.Interior
                .ColorIndex = Excelcommand
            End With
        Next irow

    End Sub

```

Display 24. Updated version of Format_VBA

Step 3: Open the DDE_Autoscript file, which will invoke the stored process to create worksheets Class and VBA_Command. Later, Format_VBA will read the sheet name, cell number and color index from VBA_Command and format the worksheet Class accordingly. See display 25 below.

	A	B	C	D	E	F
	Name	Sex	Age	Height	Weight	
0	Alfred	M	14	65	112.5	
1	Alice	F	13	56.5	84	
2	Barbara	F	13	65.3	98	
3	Carol	F	14	62.8	102.5	
4	Henry	M	14	63.5	102.5	
5	James	M	12	57.3	83	
6	Jane	F	12	59.8	84.5	
7	Janet	F	15	62.5	112.5	
8	Jeffrey	M	13	62.5	84	
9	John	M	12	59	99.5	
10	Joyce	F	11	51.3	50.5	
11	Judy	F	14	64.3	90	
12	Louise	F	12	56.3	77	
13	Mary	F	15	66.5	112	
14	Philip	M	16	72	150	

Display 25. Stored Process final output in Sheet1

AUTOMATED DDE REPLACEMENT SOLUTION FOR WORD DOCUMENT

The DDE replacement solution can be further expanded to format a Microsoft Word document. Overall the process will be the same except we will be using Microsoft Word SAS Add-in to invoke the stored process and VBA macro. Please see some high-level steps below.

Step 1: Create VBA macro Watermark using the process like the one described above for the VBA macro Format_VBA in Word. See below display 26 for the VBA macro Watermark code.

```

(General) Watermark
Sub Watermark ()
ActiveDocument.Sections(1).Range.Select
ActiveWindow.ActivePane.View.SeekView = wdSeekCurrentPageHeader
Selection.HeaderFooter.Shapes.AddTextEffect (
    PowerPlusWaterMarkObject5788307, "ASAP", "Calibri", 1, False, False, 0, 0 _
).Select
Selection.ShapeRange.Name = "PowerPlusWaterMarkObject5788307"
Selection.ShapeRange.TextEffect.NormalizedHeight = False
Selection.ShapeRange.Line.Visible = False
Selection.ShapeRange.Fill.Visible = True
Selection.ShapeRange.Fill.Solid
Selection.ShapeRange.Fill.ForeColor.RGB = RGB(192, 192, 192)
Selection.ShapeRange.Fill.Transparency = 0
Selection.ShapeRange.Rotation = 315
Selection.ShapeRange.LockAspectRatio = True
Selection.ShapeRange.Height = InchesToPoints(3.93)
Selection.ShapeRange.Width = InchesToPoints(5.24)
Selection.ShapeRange.WrapFormat.AllowOverlap = True
Selection.ShapeRange.WrapFormat.Side = wdWrapNone
Selection.ShapeRange.WrapFormat.Type = 3
Selection.ShapeRange.RelativeHorizontalPosition = _
    wdRelativeVerticalPositionMargin
Selection.ShapeRange.RelativeVerticalPosition = _
    wdRelativeVerticalPositionMargin
Selection.ShapeRange.Left = wdShapeCenter
Selection.ShapeRange.Top = wdShapeCenter
ActiveWindow.ActivePane.View.SeekView = wdSeekMainDocument
End Sub

```

Display 26. Watermark macro in Word Document

Step 2: To invoke the SAS stored process and VBA macro together add a call for Watermark within InsertStoredProcess in Word. See below display 27 for the InsertStoredProcess VBA macro code. Later, create a DDE_Autoscript for Word following similar steps mention for Excel. See display 27 below.

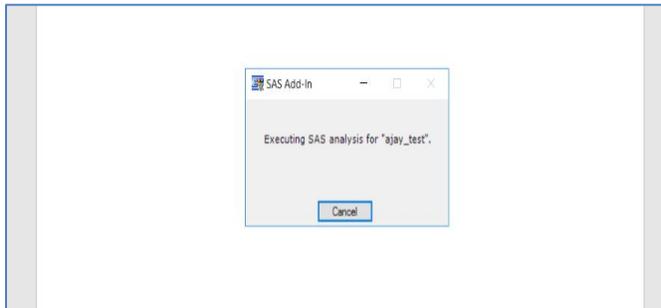
```

Sub InsertStoredProcess()
    Dim sas As SASWordAddIn
    Set sas = Application.COMAddIns.Item("SAS.WordAddIn").Object
    sas.InsertStoredProcess "/My folder/Ajay_Test"
    ' Call VBA Macro
    Call Watermark
End Sub

```

Display 27. VBA macro to execute stored process and VBA macro in Word

Step 3: Open DDE_Autoscript, which will further invoke the stored process, e.g. Ajay_Test and Watermark VBA code. See display 28 below.



Display 28. Stored Process Execution in Word

Step 4: Once the Autoscript in Word is executed then the final Word output will have the watermark. See display 29 below.

Shoe Sales By Region and Product		
Region	Product	Total Sales
Africa	Boot	\$119,835
	Men's Casual	\$562,794
	Men's Dress	\$318,500
	Sandal	\$190,409
	Slipper	\$337,076
	Sport Shoe	\$22,150
	Women's Casual	\$417,516
	Women's Dress	\$374,308
Asia	Boot	\$62,708
	Men's Casual	\$11,754
	Men's Dress	\$119,366
	Sandal	\$8,208
	Slipper	\$152,032
	Sport Shoe	\$2,092
	Women's Casual	\$25,837
	Women's Dress	\$78,234

Display 29. Final formatted output in Word.

CONCLUSION

This paper has described the replacement solution to efficiently retain the DDE functionality after migration to Grid Computing. The beauty of this solution is it is using the existing SAS and Microsoft Office technology which is easier to learn, documented, and adds no cost implication. Providing users with SAS macros to write to Excel or Word was crucial to the success of our organization’s transition to SAS grid. For our users that already were using

macros instead of DDE commands the transition was easier as they required fewer edits to their code. While DDE may have become outdated, our need to execute some of its functionality from within SAS has not gone away and is not likely to in the near future.

REFERENCES

Besse Tim, 2011. Tips and Techniques for Automating the SAS® Add-In for Microsoft Office with Visual Basic for Applications available at:

<https://www.mwsug.org/proceedings/2011/saspres/MWSUG-2011-SP01.pdf>

Zhu Yanrong, 2016. The Combination of SAS® and VBA Makes Life Easier available at:

<https://support.sas.com/resources/papers/proceedings16/7120-2016.pdf>

Kinney Saki and Wilson David, 2017. A SAS macro replacement for Dynamic Data Exchange (DDE) for use with SAS grid available at:

https://analytics.ncsu.edu/sesug/2017/SESUG2017_Paper-109_Final_PDF.pdf

Aanderud Tricia, Hall Angela, 2013. Your First SAS® Stored Process available at:

<http://support.sas.com/resources/papers/proceedings13/148-2013.pdf>

http://support.sas.com/rnd/itech/doc9/dev_guide/stprocess/program.html

<http://support.sas.com/kb/42/983.html>

ACKNOWLEDGMENTS

Thanks to Ryan Wilkins, Lindsay Dean, Lynn Mullins, Ken Borowiak, Richard DAMato, Lynn Clipstone, and PPD Management for their reviews and comments. Thanks to my family for their support.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Ajay Gupta, M.S.

PPD

3900 Paramount Parkway

Morrisville, NC 27560

Work Phone: (919)-456-6461

Fax: (919) 654-9990

E-mail: Ajay.Gupta@ppdi.com,

Ajaykailasgupta@aol.com



DISCLAIMER

The content of this paper are the works of the authors and do not necessarily represent the opinions, recommendations, or practices of PPD.

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.