

SAS® GLOBAL FORUM 2021

Paper 1088-2021

Alternate solution to %WINDOW statement in SAS® Enterprise Guide®: PowerShell

Sumit Pratap Pradhan, Syneos Health®

ABSTRACT

%WINDOW statement provides facility to define customized window that can be used to collect user input and display customized message. It is controlled by macro processor. SAS® Enterprise Guide® is a powerful windows application making SAS programming much easier and organized. Since SAS® Enterprise Guide® works on client/server model, it doesn't support %WINDOW statement. PowerShell is a powerful automation tool, consisting of command-line shell and scripting language. Windows PowerShell supports GUI programming.

On a few occasions, SAS program needs to perform task dynamically based on the inputs provided by user. Since SAS® Enterprise Guide® doesn't support %WINDOW statement, an alternate solution was needed to encounter this scenario. I came across two solutions:

1-Use Prompt Manager to create a prompt which can be used to take user input; however, the drawback is that the Prompt Manager is available only if you create a project. So it doesn't suit to the requirement.

2-Use PowerShell. It has the capability to design customized window.

So, PowerShell is selected to design form. In this paper, the entire process is explained through screenshots and PowerShell code is shared for reference.

INTRODUCTION

%WINDOW statement provides the facility to define a customized window that can be used to collect user input and display customized messages. It is controlled by macro processor. SAS® Enterprise Guide® is a powerful windows application making SAS programming much easier and organized. Since SAS® Enterprise Guide® works on client/server model, it doesn't support %WINDOW statement. PowerShell is a powerful automation tool, consisting of command-line shell and scripting language. Windows PowerShell supports GUI programming.

BACKGROUND

On few occasions, SAS program needs to perform task dynamically based on the inputs provided by user. Since SAS® Enterprise Guide® doesn't support %WINDOW statement, an alternate solution was needed to encounter this scenario. I work in Citrix environment that provides Windows Server 2012 R2 and had constraint to take user input either using SAS programming (through SAS® Enterprise Guide®) or some open source language/software that doesn't need installation by IT department. I came across two solutions:

- 1) Use Prompt Manager to create prompt which can be used to take user input but the drawback is that the Prompt Manager is available only if you create a project. So it doesn't suit to the requirement
- 2) Use PowerShell. It has the capability to design customized window from simpler ones to complex. PowerShell has following advantages:

- i) It is open-source framework consisting of command-line shell and scripting language.
- ii) It is built on .NET core.
- iii) It is pre-installed on Windows Server 2012 R2. So it requires no overhead for IT team.

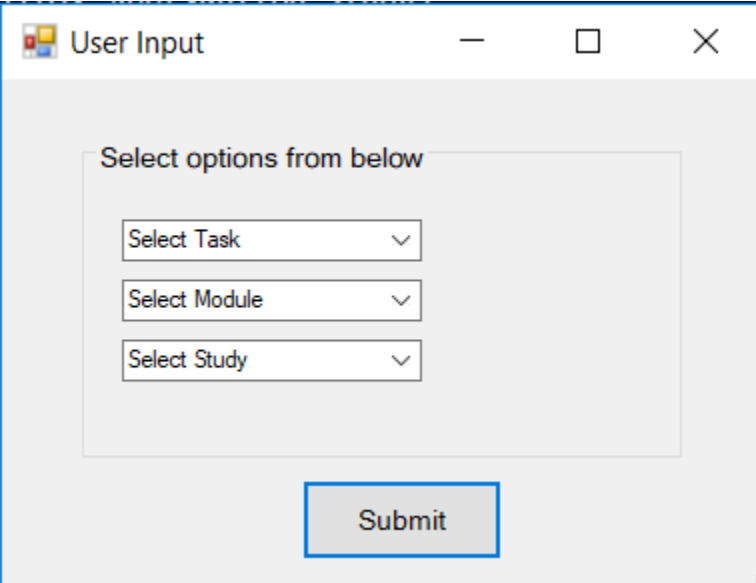
DESIGN FORM USING POWERSHELL

I have selected PowerShell to design form. Let's understand with the help of an example. Task is to create XPT or Define XML based on information provided by user. User defined SAS macro is written for XPT creation or Define XML generation. It needs the following three inputs from user:

- 1) Task to be performed (XPT creation or Define generation)
- 2) Module for which task needs to be performed (SDTM or ADaM)
- 3) Study name for which task needs to be performed – this will display list of all study available to project area and user can select any one study from the list

PowerShell script is written to take user input and save the input taken to a text file. The screen below was created using PowerShell. See Figure 1

Program is mentioned in the Appendix (Screen.ps1)



The image shows a screenshot of a Windows application window titled "User Input". The window has a standard Windows title bar with minimize, maximize, and close buttons. Inside the window, there is a light gray background. At the top, the text "Select options from below" is displayed. Below this text, there are three vertically stacked dropdown menus. The first dropdown menu is labeled "Select Task", the second is labeled "Select Module", and the third is labeled "Select Study". Each dropdown menu has a small downward-pointing arrow on its right side. At the bottom center of the window, there is a rectangular button with the text "Submit" on it.

Figure 1. User Input Form

User needs to select option from drop-down box. The screenshot below shows the options selected by user. See Figure 2

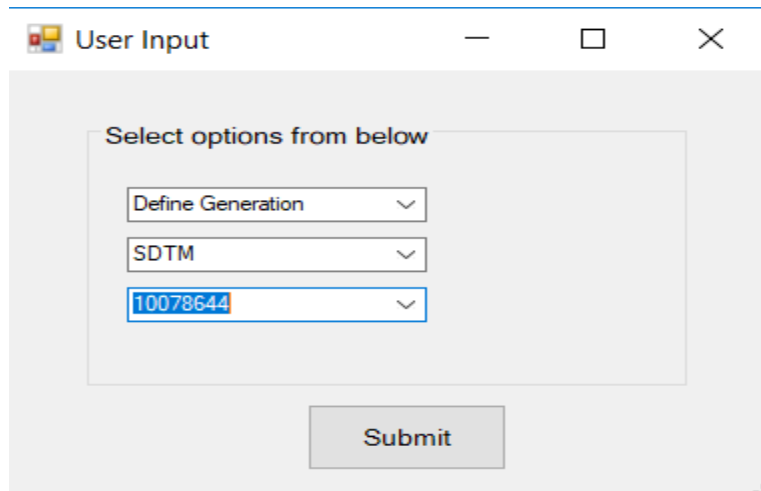


Figure 2. Options selected by User

GETTING RESULT

Now SAS program can utilize the text file to read and produce the appropriate result i.e. either create XPT or generate Define for selected module and selected study.

Consider SAS program is ready to read text file and run the user defined SAS macro for XPT creation or Define generation. Running the SAS program could be done in two ways:

- 1) Run the PowerShell Script which will generate the text file. Now run the SAS program in SAS® Enterprise Guide® or batch mode.
- 2) A more efficient way could be to write a single PowerShell script to take user input and run SAS program. Add the code to run SAS program at the end of the script. This is better approach because programmer only needs to run the PowerShell script and it will produce the intended result.

CONCLUSION

In this paper, only one aspect of PowerShell has been discussed although PowerShell is very powerful automation tool that can be utilized for various purposes. Combining SAS with PowerShell gives more flexibility to programmers to automate tasks.

REFERENCES

1. SAS Documentation
https://documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.5&docsetId=mcr_olref&docsetTarget=n0ojc9rtfrzvbdn16kqd2c5mzobq.htm&locale=en
2. PowerShell Documentation: Overview
<https://docs.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7.1>
3. PowerShell Documentation: Creating a Custom Input Box
[https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-powershell-1.0/ff730941\(v=technet.10\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-powershell-1.0/ff730941(v=technet.10)?redirectedfrom=MSDN)

CONTACT INFORMATION

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

Sumit Pratap Pradhan
Syneos Health, Principal Statistical Programmer
Building No. 14, Tower B, DLF Cyber City, Gurgaon - 122002, Haryana, India
E-mail: sumit.pradhan@syneoshealth.com
LinkedIn: <https://www.linkedin.com/in/sumit-pradhan-71133345/>

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

APPENDIX

```
<# ***** POWERSHELL SCRIPT *****  
  
***** SCREEN.PS1 *****  
  
***** #>  
  
#Load Required assemblies for Form Creation  
Add-type -AssemblyName System.Windows.Forms  
  
#Design User Input Form  
$Form_Input = New-Object System.Windows.Forms.Form  
$Form_Input.Text='User Input'  
$Form_Input.width = 400  
$Form_Input.height = 300  
  
#Add Group  
$Group_Input = New-Object System.Windows.Forms.GroupBox  
$Group_Input.Location=New-Object System.Drawing.Size(40,30)  
$Group_Input.Size=New-Object System.Drawing.Size(300,160)  
$Group_Input.Text='Select options from below'  
  
#Add Drop-Down Button having Task details  
$Button_Task = New-Object System.Windows.Forms.ComboBox  
$Button_Task.Location=New-Object System.Drawing.Size(20,40)  
$Button_Task.Name='Button_Task'  
$Button_Task.Size=New-Object System.Drawing.Size(150,20)  
$Button_Task.Text='Select Task'  
#Add to Group  
$Group_Input.Controls.Add($Button_Task)  
  
#Add Drop-Down for Module  
$Button_Module = New-Object System.Windows.Forms.ComboBox  
$Button_Module.Location=New-Object System.Drawing.Size(20,70)  
$Button_Module.Name='Button_Module'  
$Button_Module.Size=New-Object System.Drawing.Size(150,20)  
$Button_Module.Text='Select Module'  
#Add to group  
$Group_Input.Controls.Add($Button_Module)
```

```

#Add Drop-Down for Study
$Button_Study = New-Object System.Windows.Forms.ComboBox
$Button_Study.Location=New-Object System.Drawing.Size(20,100)
$Button_Study.Name='Button_Study'
$Button_Study.Size=New-Object System.Drawing.Size(150,20)
$Button_Study.Text='Select Study'
#Add to group
$Group_Input.Controls.Add($Button_Study)

# TASK VALUES
$Button_Task.Items.Clear()
ForEach ($Itemt in 'XPT Creation', 'Define Generation') {
    $Button_Task.Items.Add($Itemt)
}

# MODULE VALUES
$Button_Module.Items.Clear()
ForEach ($Itemm in 'SDTM', 'ADaM') {
    $Button_Module.Items.Add($Itemm)
}

# STUDY VALUES
$ROOTPATH='C:\Users\supradha\Desktop\SAS';
$areaval = (Get-ChildItem -Path $ROOTPATH -Directory -Exclude root).Name
$Button_Study.Items.Clear();
ForEach ($Itemts in $areaval) {
    $Button_Study.Items.Add($Itemts)
}

#Add Submit Button
$Button_Submit = New-Object System.Windows.Forms.Button
$Button_Submit = New-Object System.Windows.Forms.Button
$Button_Submit.Location=New-Object System.Drawing.Size(150,200)
$Button_Submit.Size=New-Object System.Drawing.Size(100,40)
$Button_Submit.Text='Submit'
$Button_Submit.DialogResult=[System.Windows.Forms.DialogResult]::abort
#Add Submit Button to Form
$Form_Input.Controls.Add($Button_Submit)

#Add Group to Form
$Form_Input.Controls.Add($Group_Input)
$Form_Input.CancelButton = $Button_Submit

$Form_Input.Add_Shown({$Form_Input.Activate()})
$Form_Input.ShowDialog()

#Create TEXT file to store selected option
"TASK: "+ $Button_Task.text +
" `r`nMODULE: "+ $Button_Module.text +
" `r`nSTUDY: "+ $Button_Study.text +
"| Out-File "$ROOTPATH\User_input.txt" -Force

```