Technical Paper

Using SAS® Studio to Open SAS® Enterprise Guide® Project Files

(Experimental in SAS Studio 3.6)

Release Information Content Version: 1.0 March 2017

Author Jennifer Jeffreys-Chen

ABSTRACT

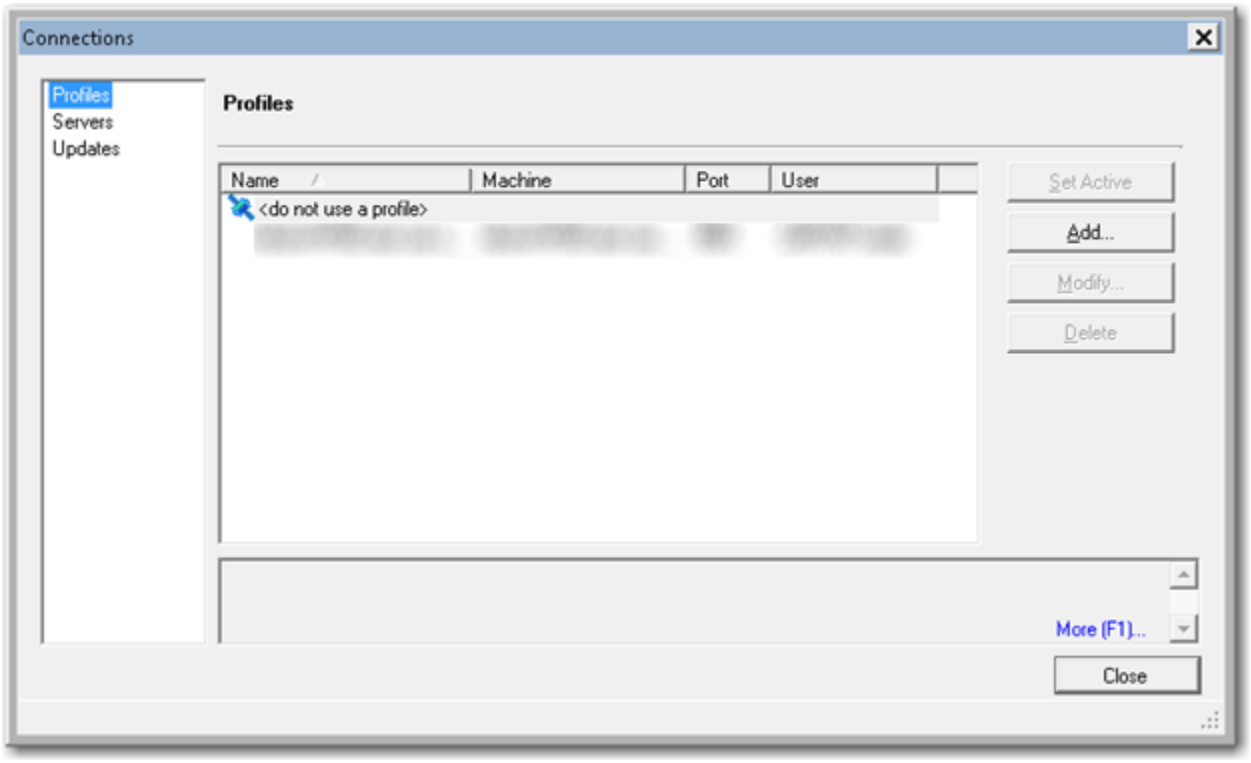
As a SAS® programmer, how often does it happen that you would like to open your SAS® Enterprise Guide® project in SAS® Studio? SAS Studio has a new feature that enables you to do just that! This paper covers how SAS Studio extracts SAS Enterprise Guide process flows from a SAS Enterprise Guide project file and converts the nodes to elements in a SAS Studio process flow.

INTRODUCTION

SAS Studio users have been requesting support for SAS Enterprise Guide projects since the introduction of the Visual Programming Perspective and SAS Studio process flows. SAS Studio 3.6 introduces an experimental feature that extracts process flows from a SAS Enterprise Guide project file (also referred to as an EGP file in this document) and converts the process flow to a SAS Studio process flow. Elements that are not supported by SAS Studio are either converted to different node types or are omitted from the process flow. A Conversion Report is provided to explain what happened to the nodes in the conversion process. This paper will help users understand how elements are converted and when a converted process flow might require manual intervention before it will run. SAS Studio Administrators must enable this feature since it is off by default. Users of SAS University Edition will find that this functionality is always enabled.

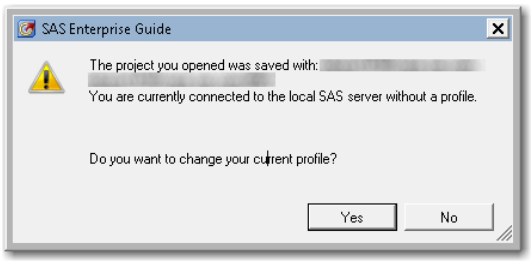
SERVER CONTEXT

Server context is very important for EGP files. In SAS Enterprise Guide, the active connection is saved in the project metadata . (To view the active connection in SAS Enterprise Guide, select **Tools** -> **Connections**. In the Connections dialog box, select **Profiles**).



Display 1 – Connections Dialog Box in SAS Enterprise Guide

If you try to open an EGP file in SAS Enterprise Guide that has a different connection than the current active connection, SAS Enterprise Guide shows a message.



Display 2 – Message for Mismatched Connections

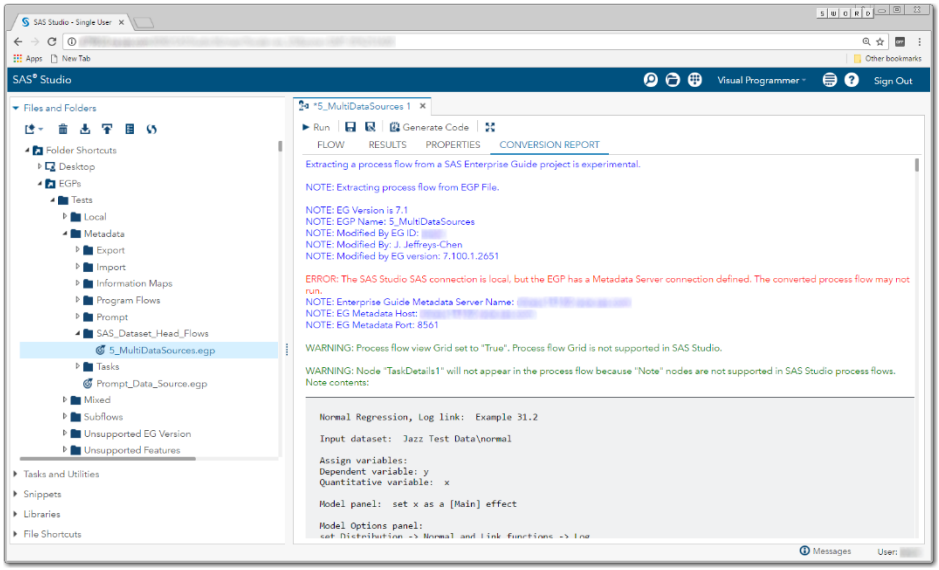
Furthermore, the data and code nodes in a SAS Enterprise Guide process flow can be associated with different connections. These connections are available to the user when constructing the process flow in SAS Enterprise Guide.

Since SAS Enterprise Guide is a desktop application, it has easy access to the local file system and SAS server. You can also access any workspace server environments defined in the SAS Metadata Server that your SAS Enterprise Guide session is configured to use.

In contrast, SAS Studio is a web interface that can have just one SAS server configuration at a time. The Single-User edition of SAS Studio has access only to the local file system and the SAS installation on the desktop. The Basic edition of SAS Studio has access only to the single workspace server that it is configured to use and the file system available to that server. Local files must be uploaded for use in the Basic edition. Finally, the Enterprise edition of SAS Studio can have multiple workspace servers defined in the SAS Metadata Server that it is configured to use. However, a user can connect to only one workspace server at a time within SAS Studio.

It is important to understand these differences when opening an EGP file in SAS Studio. If the environment for the SAS Enterprise Guide project and the environment for SAS Studio do not match or if the data or executable nodes in the EGP do not match the SAS Studio environment where the conversion occurs, the conversion report will provide warnings about the mismatches.

The following display shows an example of the type of message that you will see if your EGP environment metadata does not match your SAS Studio environment.

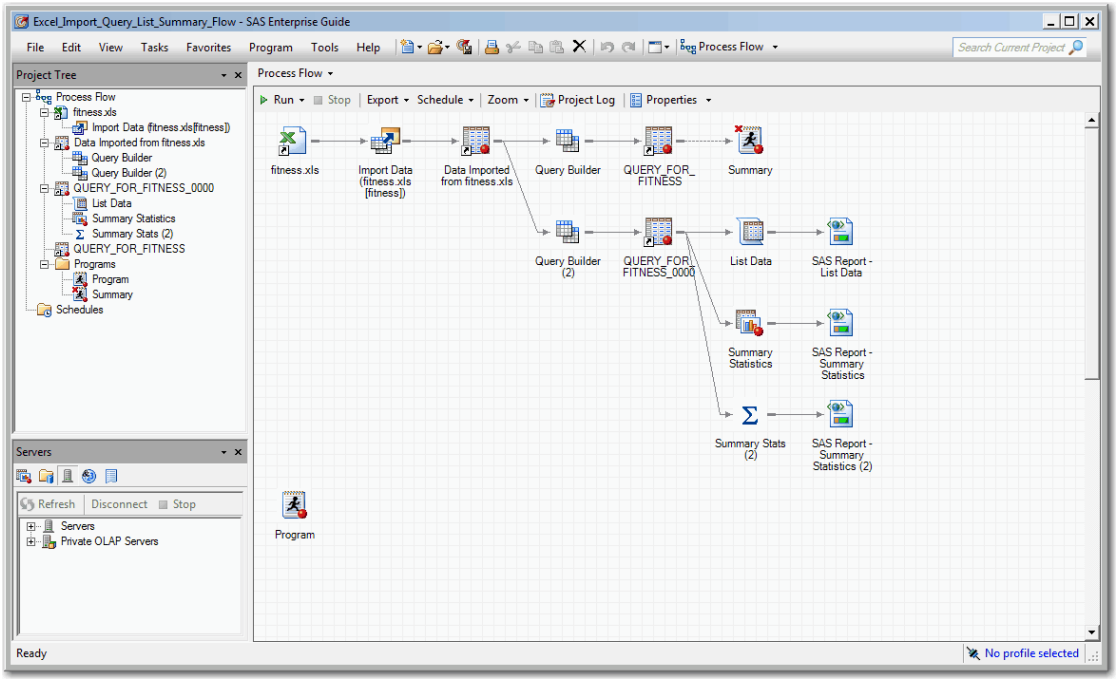


Display 3 - Mismatched SAS Studio and EGP Environments

OPENING SAS ENTERPRISE GUIDE PROJECT FILES

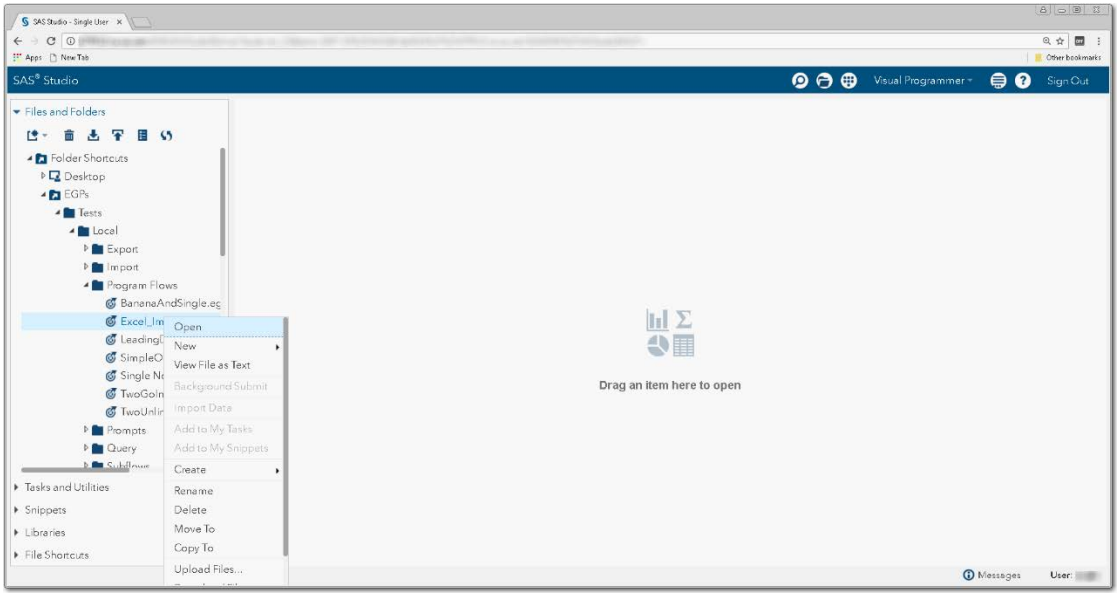
Opening an EGP file in SAS Studio results in a new Process Flow tab that contains the converted process flows. A **CONVERSION REPORT** tab indicates how EGP nodes were converted, and notes any potential problems with the converted SAS Studio process flow (.cpf file).

Here is a process flow displayed in SAS Enterprise Guide.



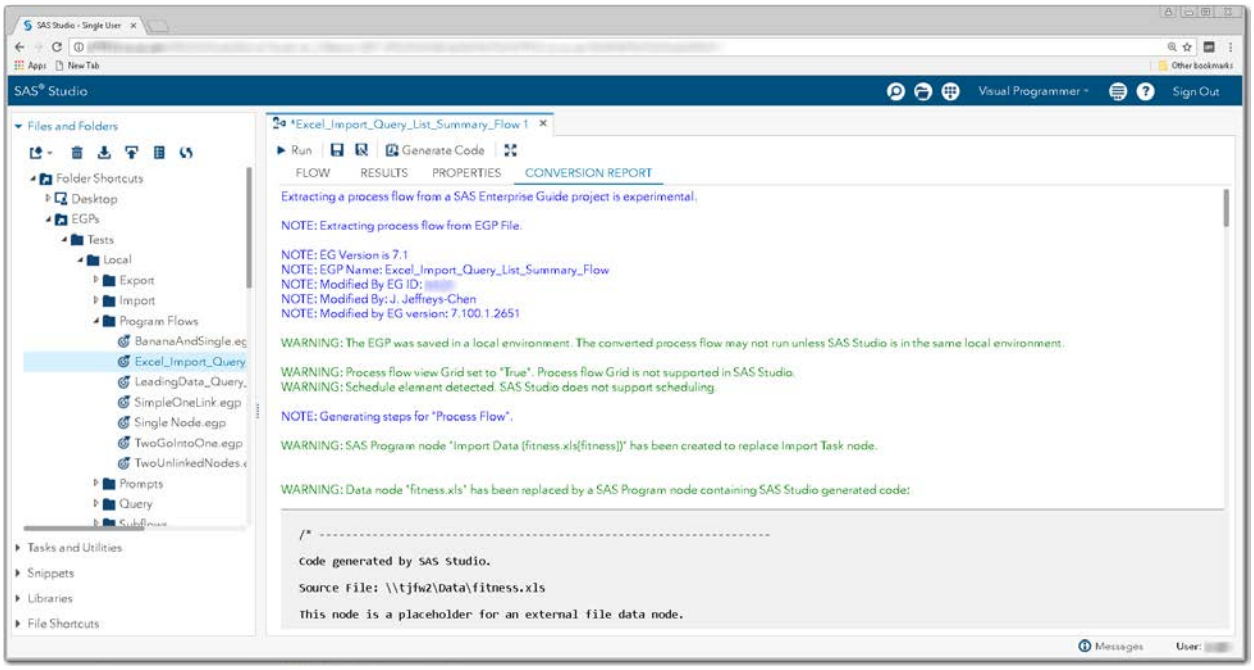
Display 4 – Process Flow in SAS Enterprise Guide

To open the EGP file in SAS Studio, open the Visual Programmer perspective in SAS Studio. Traverse to the directory that contains the EGP file. To open the file, either double click on the file or select **Open** from the pop-up menu.



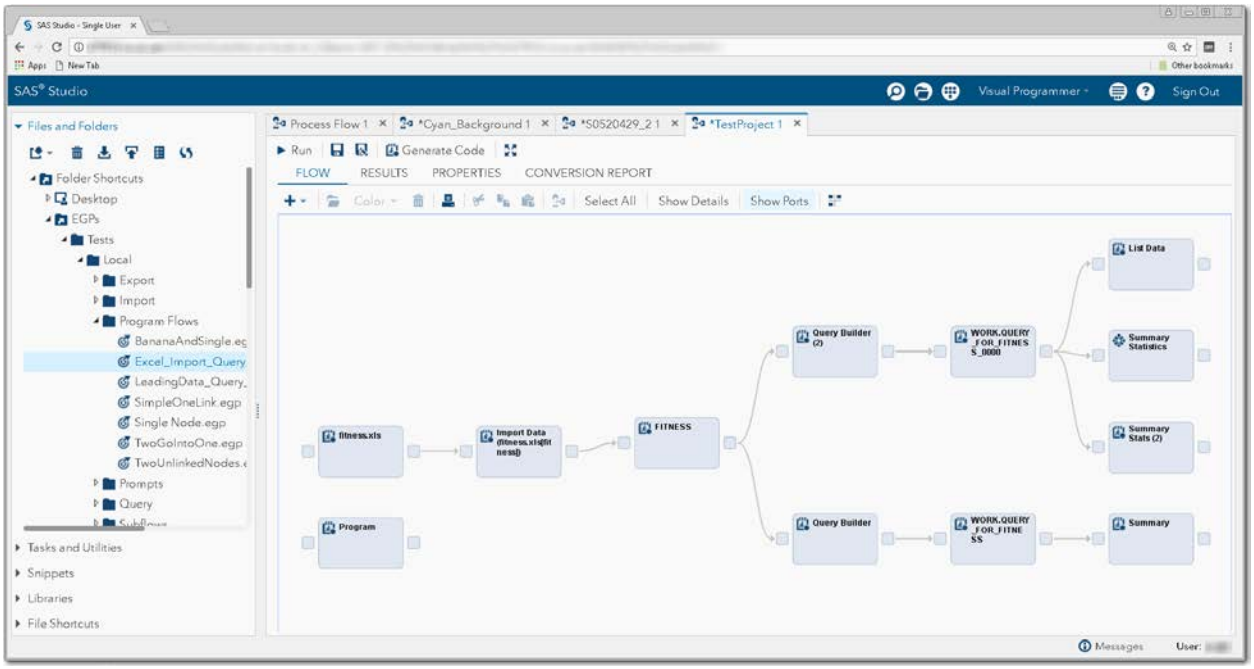
Display 5 - Opening an EGP File in SAS Studio

SAS Studio displays a process flow tab with the **CONVERSION REPORT** tab selected.



Display 6 – Contents of CONVERSION REPORT Tab

The converted flow appears in the **FLOW** tab. This display shows the converted process flow from Display 4.



Display 7 – A SAS Enterprise Guide Project in SAS Studio

You can save the converted flow to a SAS Studio process flow (.cpf) file, and the conversion report will persist with it. It is important to understand that the conversion report is static and will not reflect any changes made to the converted process flow.

The following processing occurs when an EGP file is converted:

- The EGP file is "unpacked" into a temporary directory.
- The project and its process flow nodes are analyzed to determine if the server context is consistent with the SAS Studio context where the EGP is being opened. Mismatched SAS connection contexts will result in conversion warnings.
- Where possible, nodes in the SAS Enterprise Guide process flow are converted to SAS Studio nodes.
- Multiple process flows in the EGP file become sub-flows in a single SAS Studio process flow.
- Prompts are replaced by generated macro code in SAS Studio. This macro code is added to the beginning of the code that depends on the prompts.
- The "unpacked" temporary directory is deleted unless the debug functionality is enabled in SAS Studio.

SUPPORTED NODES

Assuming that the environments for the EGP file and SAS Studio match, here are the node conversions (from SAS Enterprise Guide to SAS Studio) that you can expect.

CONVERTED

- Program
- SAS Studio Tasks

CONVERTED TO OTHER NODE TYPES

- Data
- SAS Enterprise Guide Tasks
- Query Builder
- Import Data
- Export File

NOT CONVERTED

- Note
- Stored processes
- OLAP cubes
- Log
- Project log
- ODS result

CONVERSION REPORT

The **CONVERSION REPORT** tab for a converted process flow provides information about the conversion process.

Here are sections in the report:

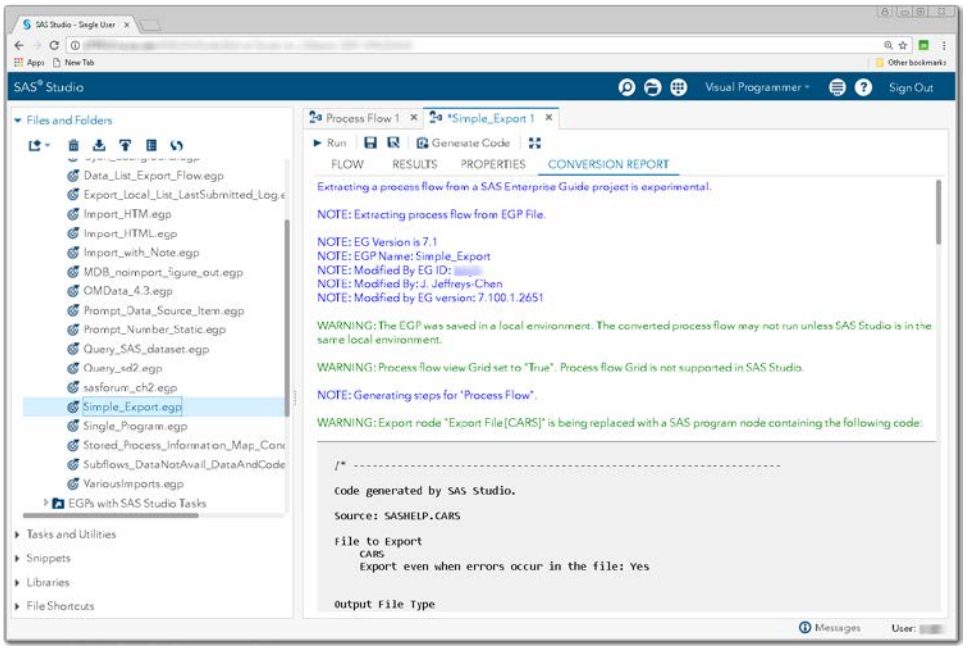
- an EGP environment summary
- a conversion log
- a node summary

ENVIRONMENT SUMMMARY

The Environment Summary provides information about the environment associated with the EGP. Attributes in this section include the following information, which is extracted directly from the EGP metadata.

- EG Version – release of SAS Enterprise Guide where the EGP was created
- EGP Name – the name of the EGP
- Modified by EG ID – the user ID of the last person to modify the EGP
- Modified by – the name of the last person to modify the EGP
- Modified by EG version – the release of SAS Enterprise Guide where the EGP was last saved

If your SAS connection in SAS Studio does not match the SAS connection in SAS Enterprise Guide where the EGP was last saved, you will get an error. If the connections might not match, you get a warning. Here is an example of when you might get a warning. You are using the Single-User edition of SAS Studio, and the EGP was saved on a desktop installation of SAS Enterprise Guide. If these local connections are not the same, there could be a problem. It is very important to make sure that the connections in SAS Studio and SAS Enterprise Guide are the same in order for the converted SAS Studio process flow to be correct.



Display 8 - Environment Summary in the Conversion Report

The Environment Summary can also contain general warnings about the SAS Studio and SAS Enterprise Guide process flows. For example in the previous display, there is a warning stating that the process flow grid in SAS Enterprise Guide is not supported in SAS Studio.

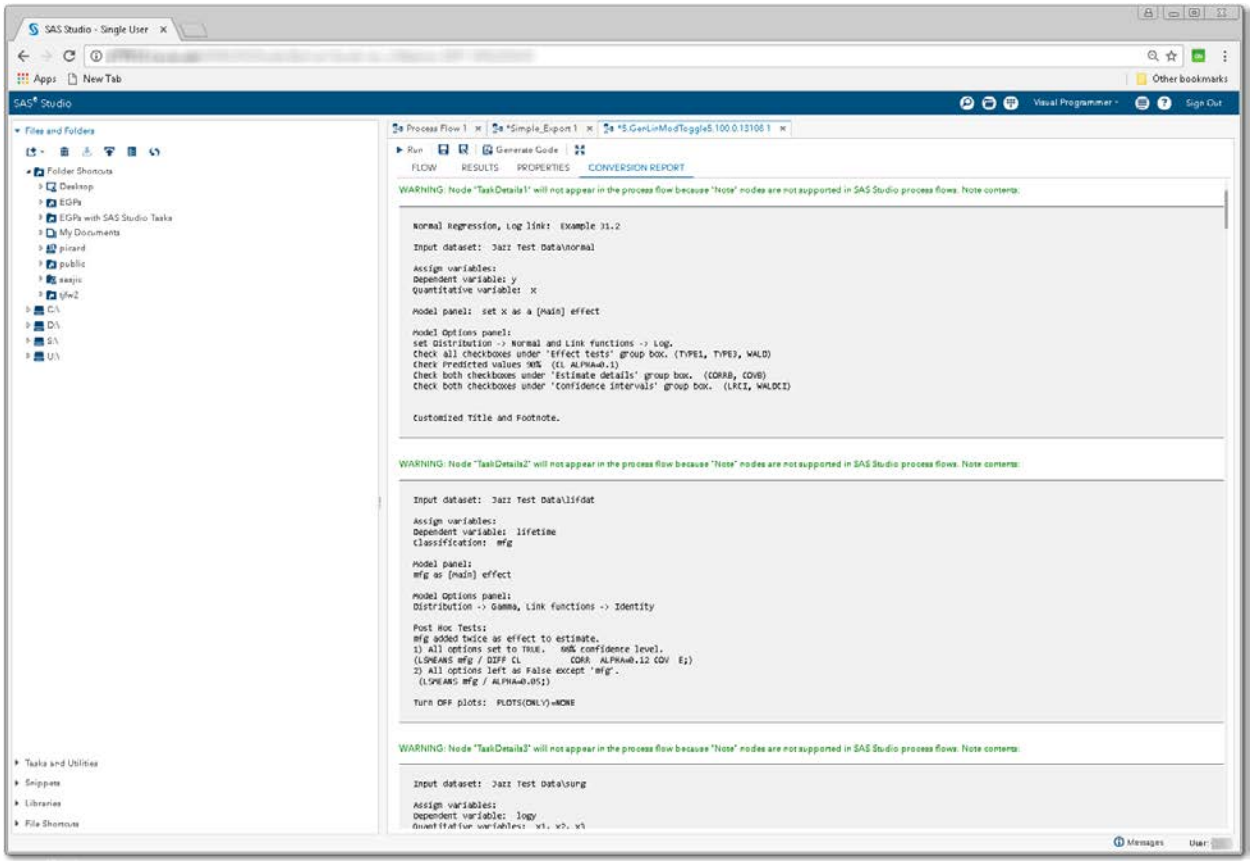
CONVERSION LOG

The conversion log shows the contents of any Note nodes from the SAS Enterprise Guide process flow as well as a running summary of the conversion activity. If you see errors in the report, your converted process flow probably will not run with the SAS connection that you are using in SAS Studio and will require manual intervention.

Note Contents

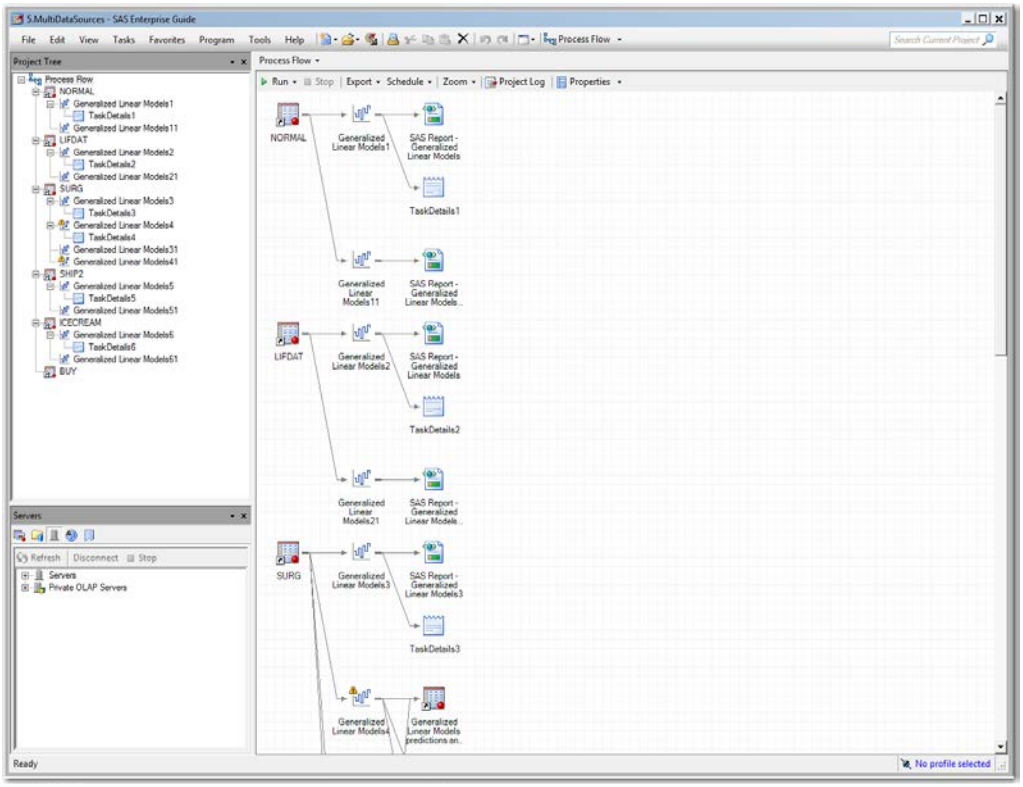
Since SAS Studio process flows do not support Note nodes, they will not appear in converted process flows. After the Environment Summary**,** there is a section that contains the contents of any Note nodes that have been eliminated.

The following example shows Note contents from the Sample Process Flow.

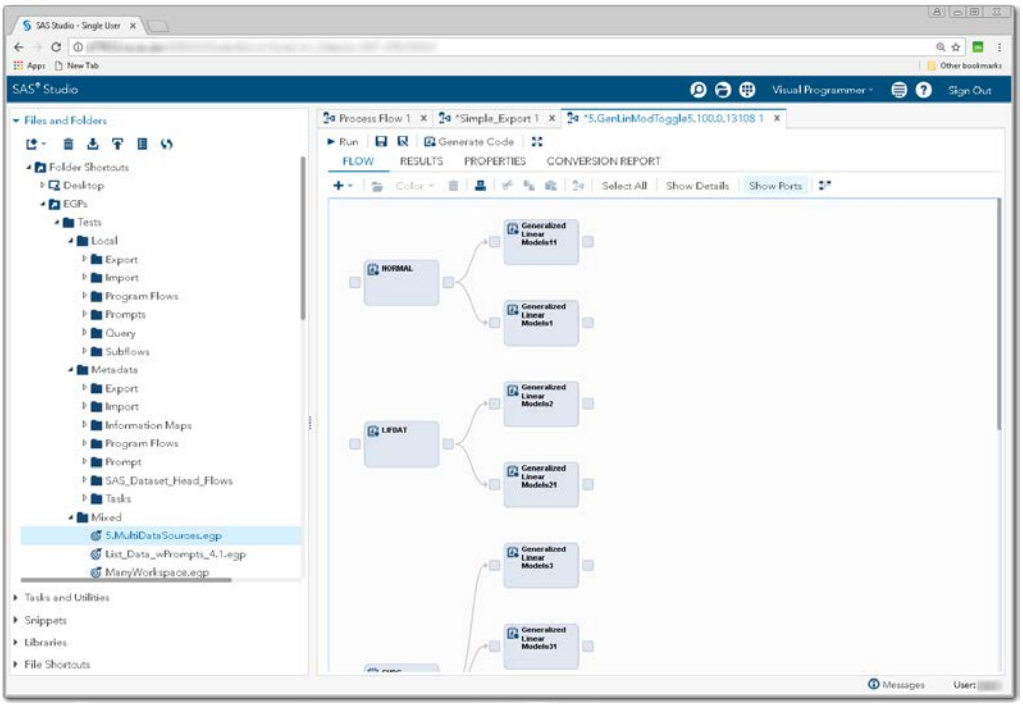


Display 9 - Contents of Eliminated Note Nodes

Sample Process Flow



Display 10 - Sample Process Flow in SAS Enterprise Guide

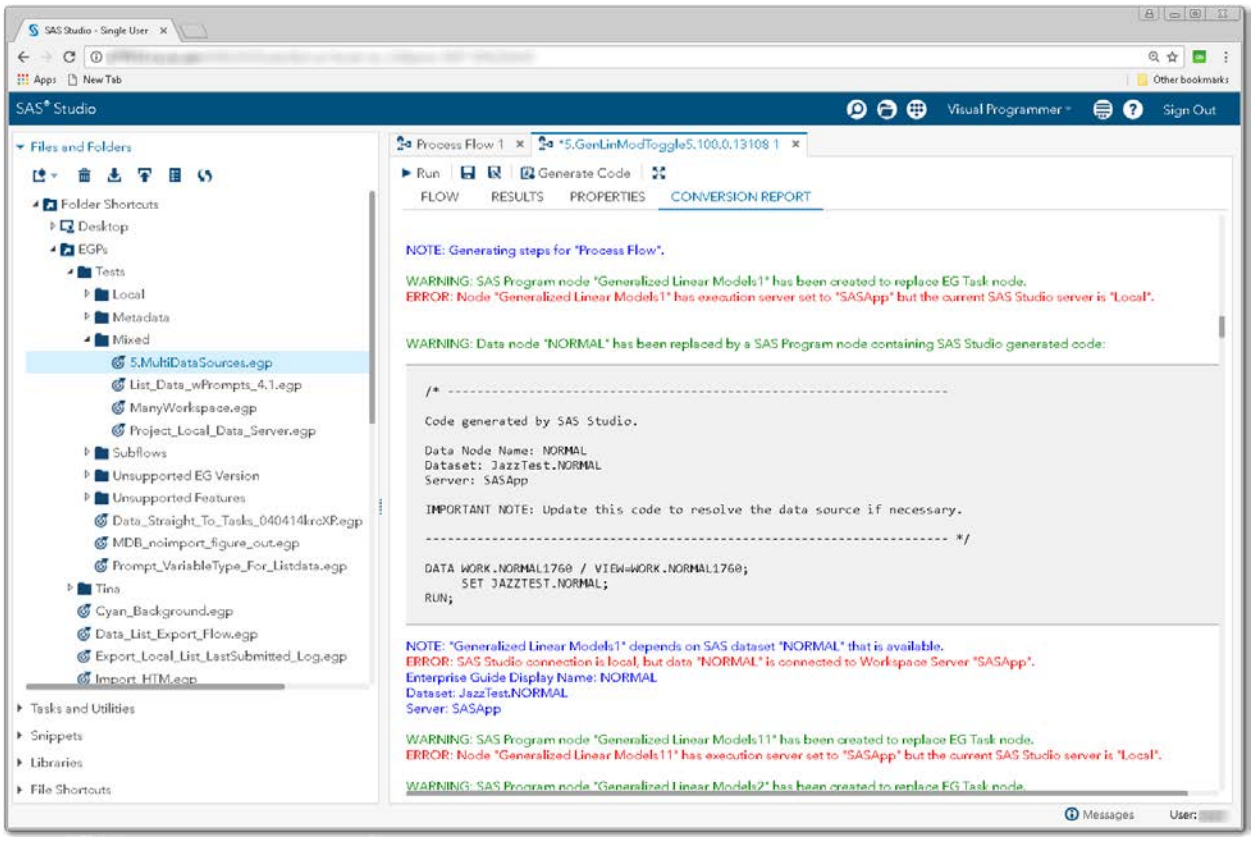


Display 11 - Sample Process Flow in SAS Studio

Generating Steps, Node Connectors, and Node Links

After the Note section of the Conversion Report, there is a running log of the conversion of the steps, ports, and connectors. Any warnings or errors for these items are noted.

In the following example, there are environment errors for SAS Enterprise Guide Task and Data nodes, conversion warnings for the tasks, and a conversion warning for a Data node.

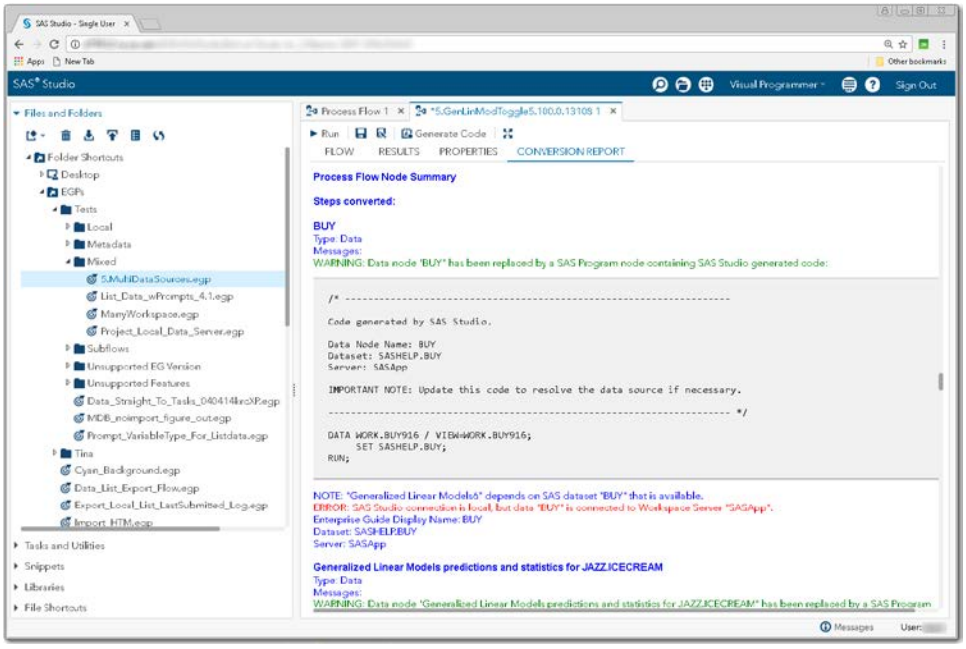


Display 12 – Running Log of Conversion Steps

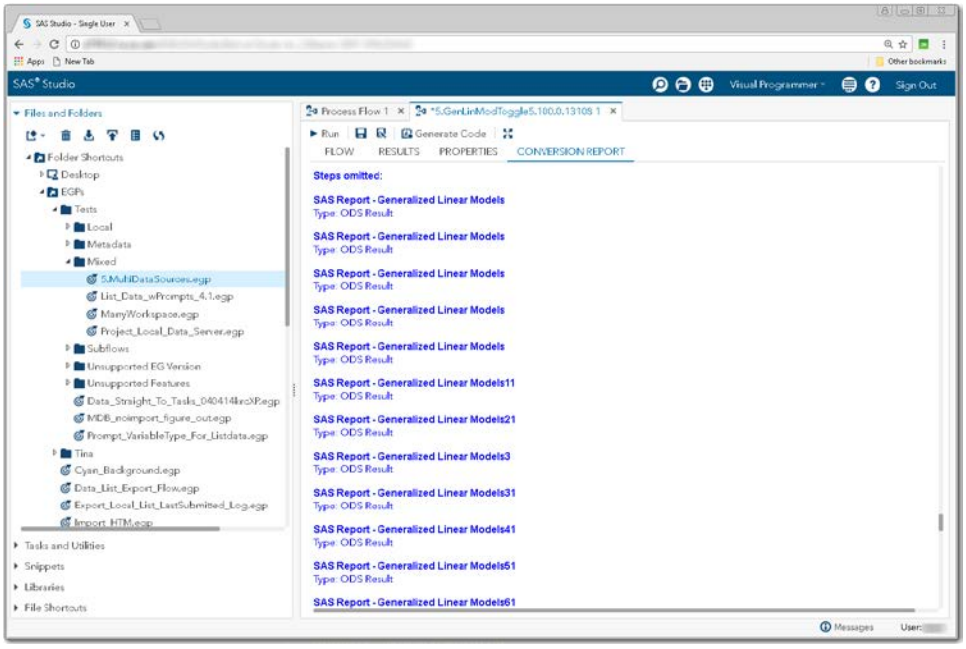
NODE SUMMARY

The Node Summary section of the Conversion Report provides a list of nodes that are included in the converted process flow and a list of nodes that are not included. Both lists are in alphabetical order. The node names are in bold, and any errors and warnings associated with the nodes are listed below the node name.

The following displays show Node Summary information for the Sample Process Flow.



Display 13 – Node Summary Showing Converted Nodes



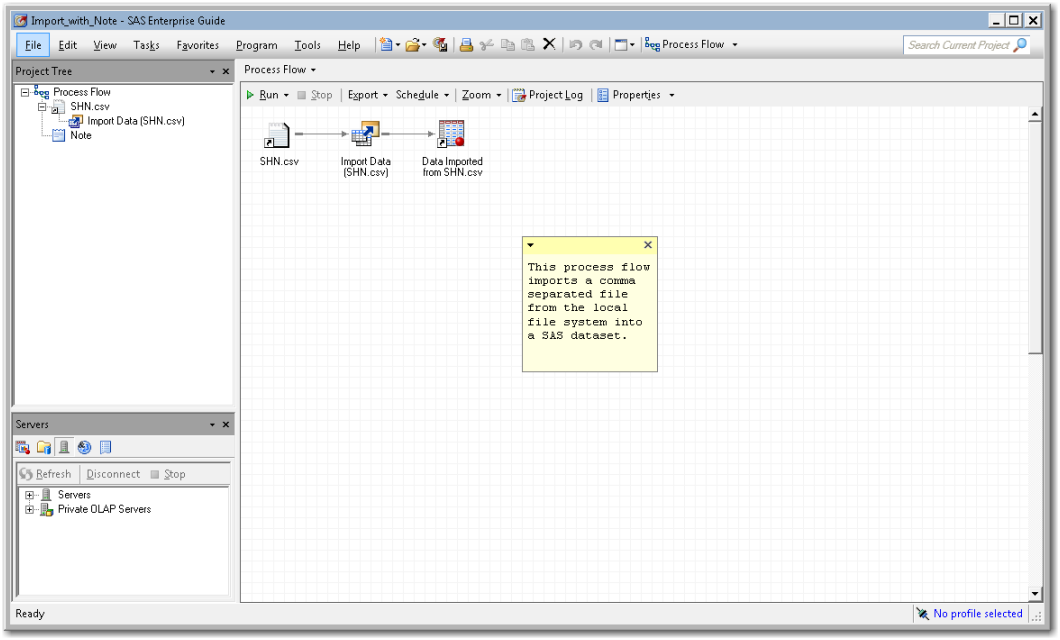
Display 14 - Node Summary Showing Nodes That Could Not Be Converted

ELEMENTS

NOTE NODES

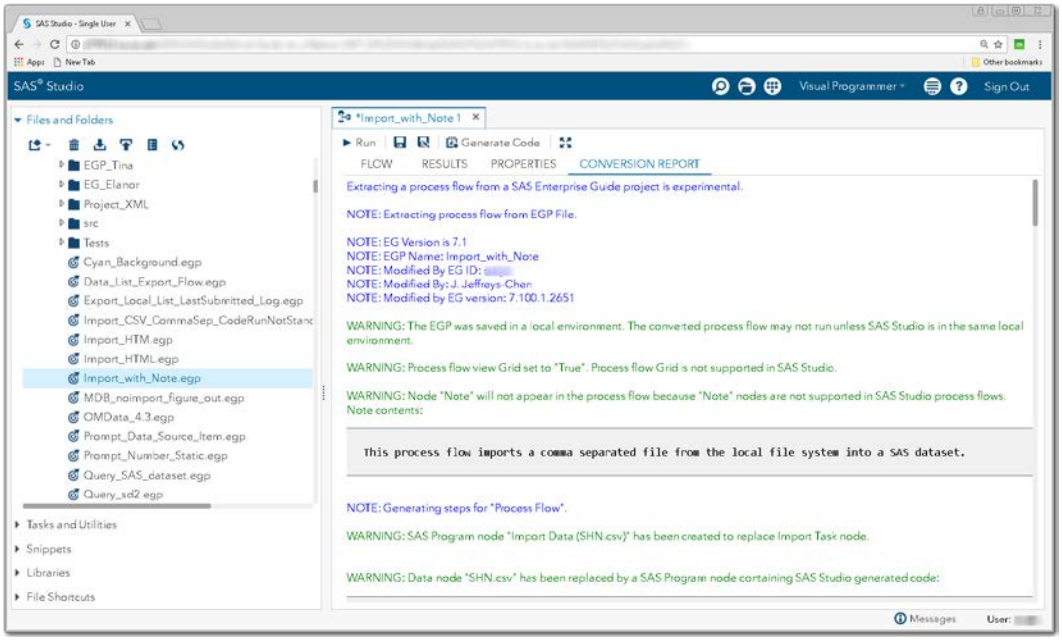
SAS Studio does not support Note nodes, so Note elements do not appear in a converted flow. The contents of the Note elements are written to the Conversion Report.

Here is an example of a Note node in a process flow in SAS Enterprise Guide.



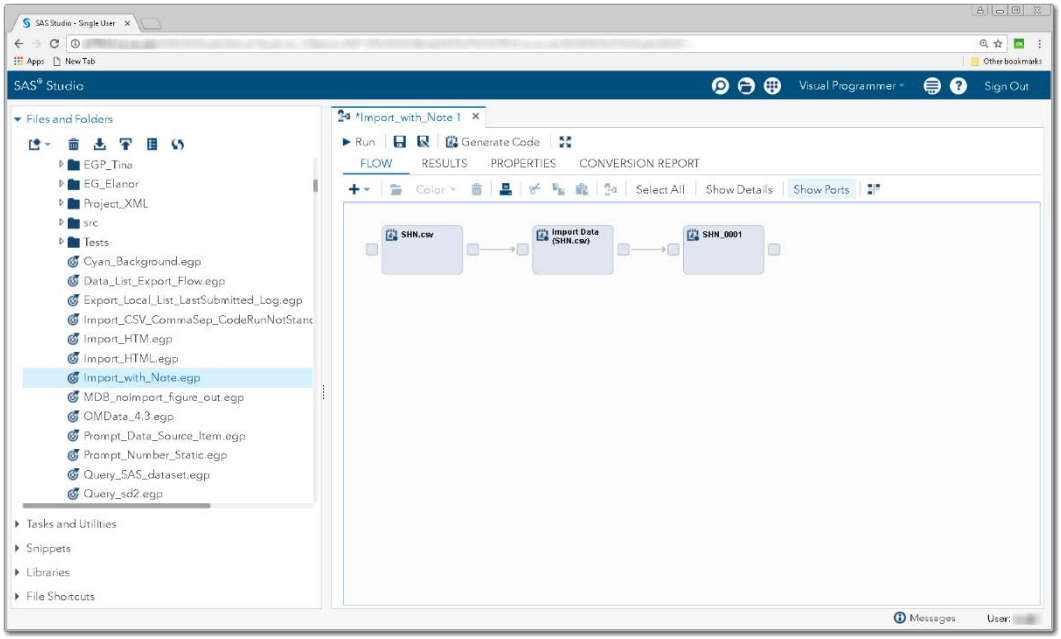
Display 15 - Note Node in Process Flow in SAS Enterprise Guide

When you open the same EGP in SAS Studio, you get a message notifying you that the Note node will not appear in the converted process flow because Note nodes are not supported in SAS Studio. The contents of the Note node is shown in the Conversion Report in the following display.



Display 16 – Note Node Warning and Contents in SAS Studio Conversion Report

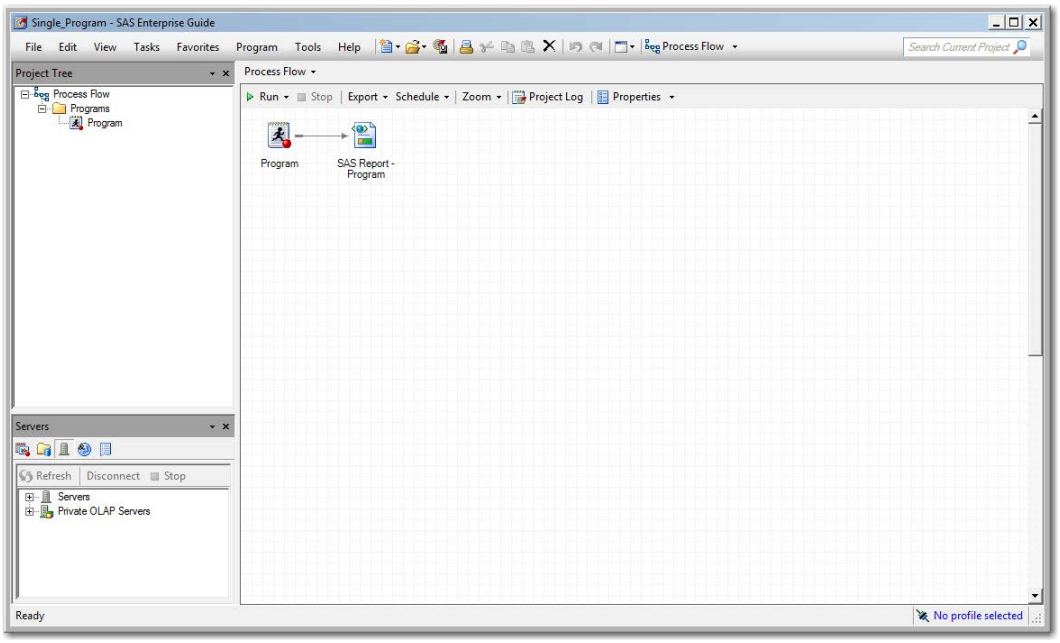
On the **FLOW** tab, you see that there is no Note node.



Display 17 - Note Node Is Not in SAS Studio Process Flow

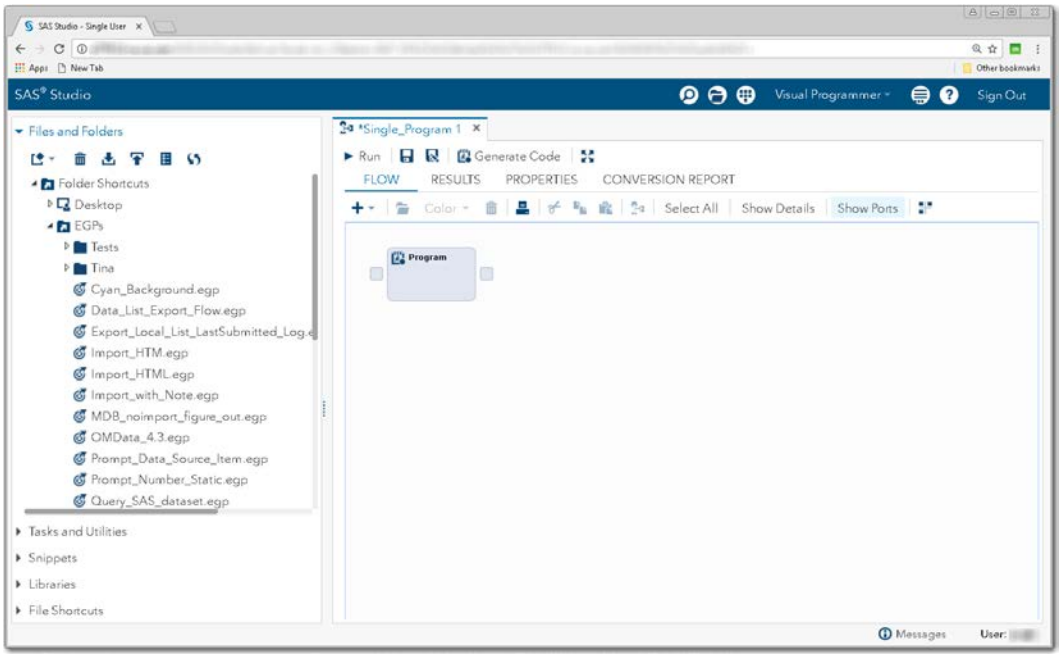
PROGRAM NODES

Program nodes are converted directly to SAS Program nodes in SAS Studio. If the contents of the Program node are stored in a file, the file must be available in order for the contents of the file to be read into converted Program node. An error is written to the Conversion Report if the file containing the program cannot be found. SAS Report nodes generated by Program nodes are not supported in SAS Studio process flows and will not appear in the converted process flow.



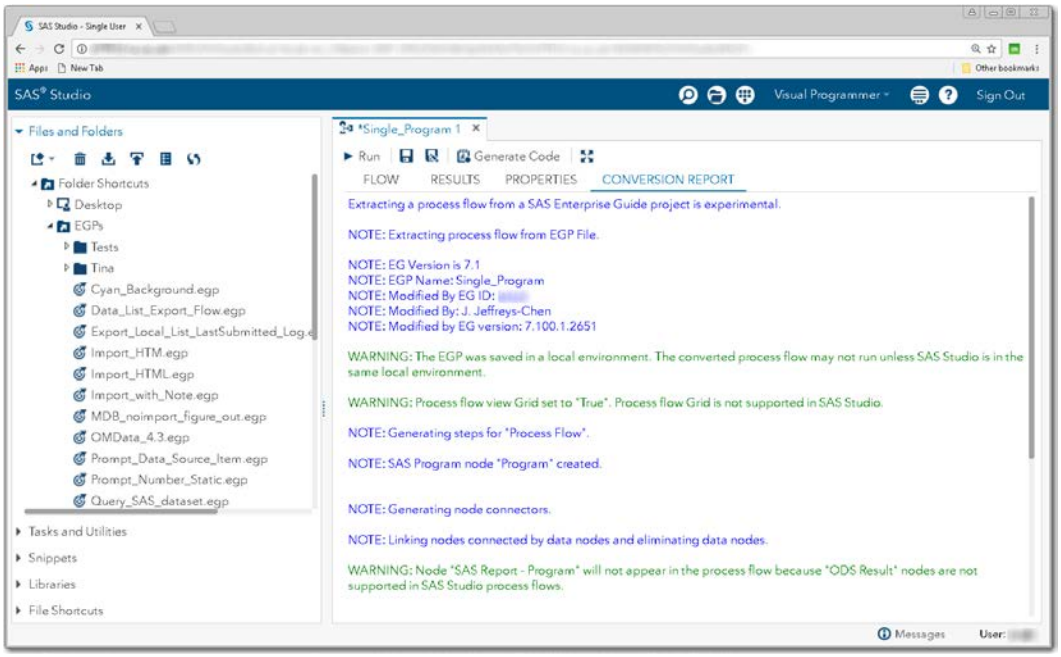
Display 18 – Program Node and SAS Report Node in SAS Enterprise Guide

Here is the converted process flow in SAS Studio. Note that the SAS Report node is not present.



Display 19 - SAS Studio Process Flow with Program Node But No SAS Report Node

There is a warning in the Conversion Report about the removal of the SAS Report node.



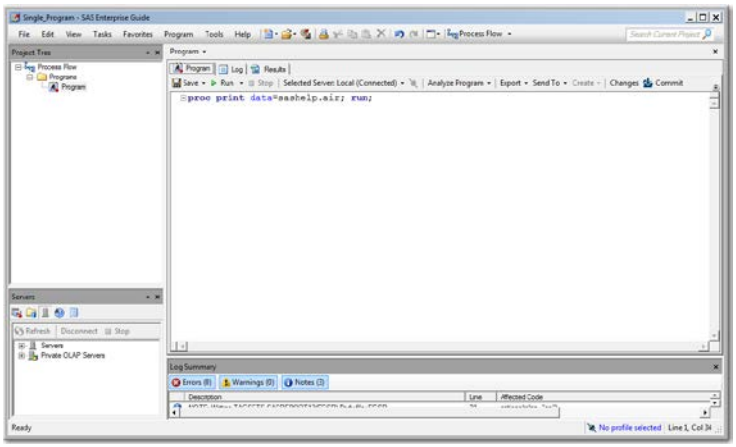
Display 20 - Warning about Removal of SAS Report Node

In SAS Enterprise Guide, Program nodes are associated with a particular SAS connection. If the SAS Studio connection does not match the connection associated with the Program node, an error appears in the log.

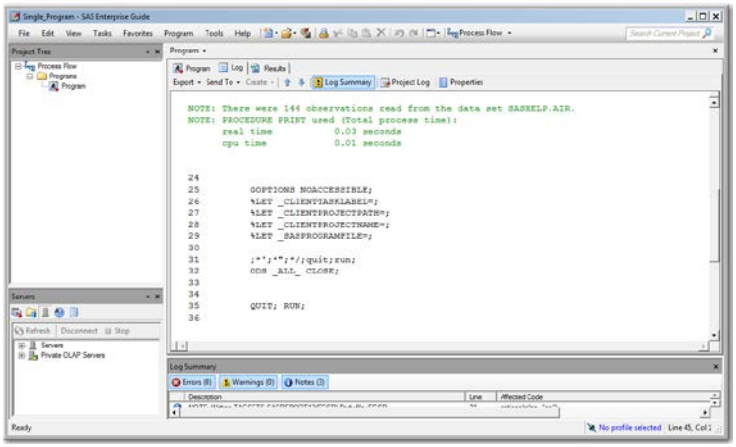
The contents of the Program node in SAS Enterprise Guide and the converted Program node in SAS Studio are very similar, as shown in the following displays.

Program Node in SAS Enterprise Guide

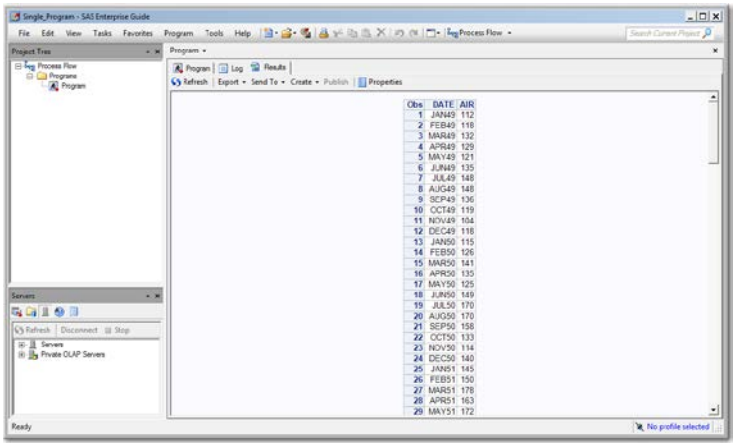
Contrast with SAS Program Node in SAS Studio.



Display 21 – Contents of Program Node in SAS Enterprise Guide



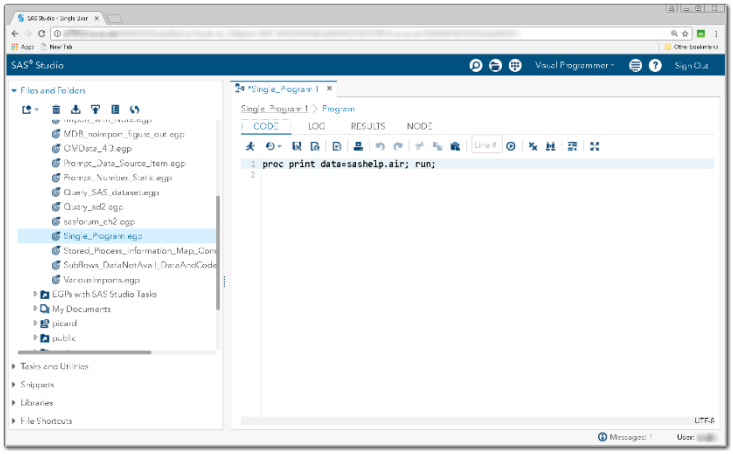
Display 22 – Log Generated by Running the Program Node in SAS Enterprise Guide



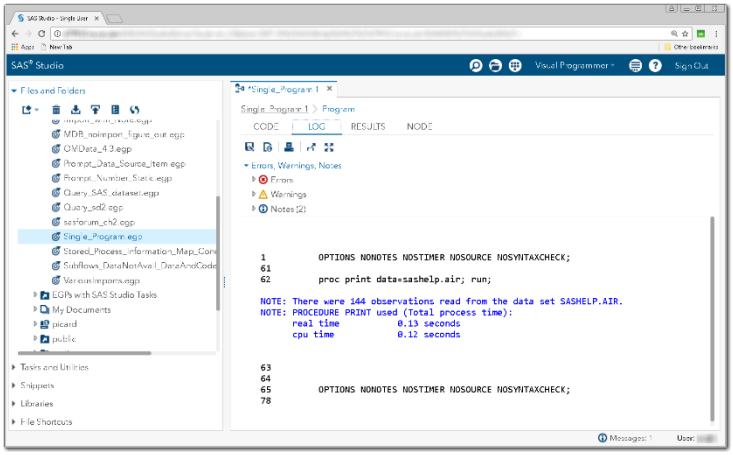
Display 23 – Results Generated by Running the Program Node in SAS Enterprise Guide

SAS Program Node in SAS Studio

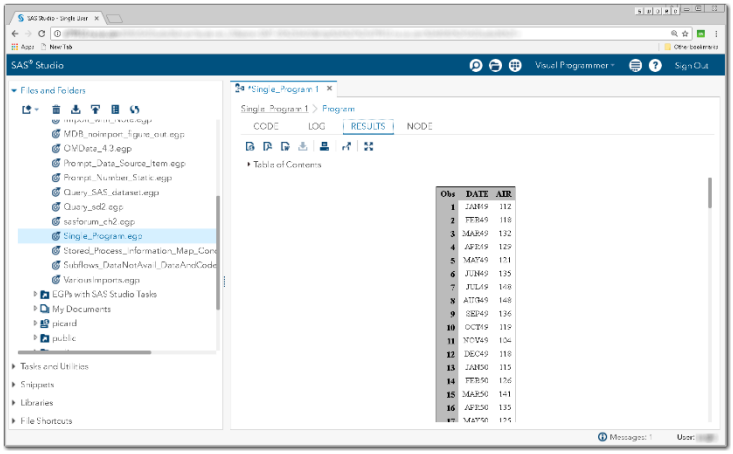
Contrast with Program Node in SAS Enterprise Guide.



Display 24 – Contents of SAS Program Node in SAS Studio



Display 25 – Log Generated by Running the SAS Program Node in SAS Studio



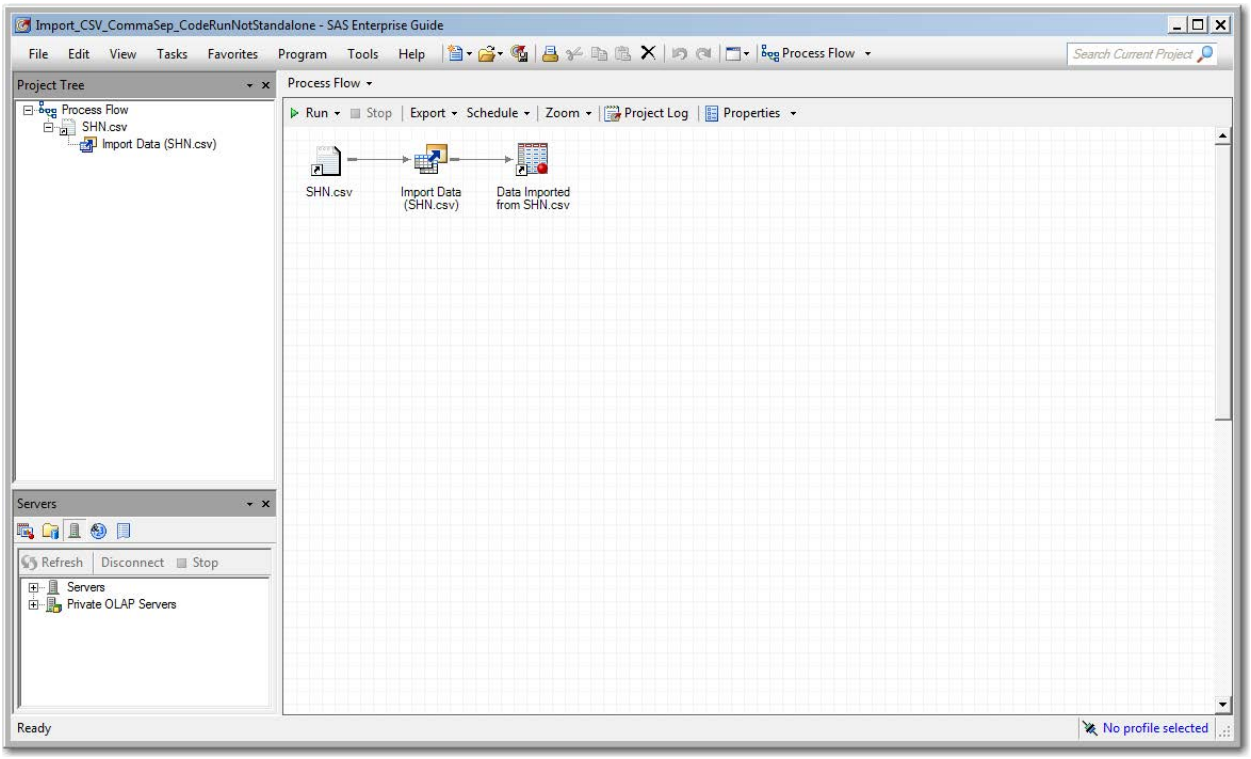
Display 26 – Results Generated by Running the SAS Program Node in SAS Studio

IMPORT DATA NODES

Because an Import Data node in SAS Enterprise Guide does not port directly to an Import Data task in SAS Studio, SAS Studio makes quite a few changes to Import Data nodes to support importing files.

SAS Enterprise Guide

In the following example, a .csv file is dropped into a SAS Enterprise Guide project, and the Import Data Wizard is used to specify the output data location, delimiter information, fields to import, and other advanced options. The result is three nodes in the SAS Enterprise Guide process flow.

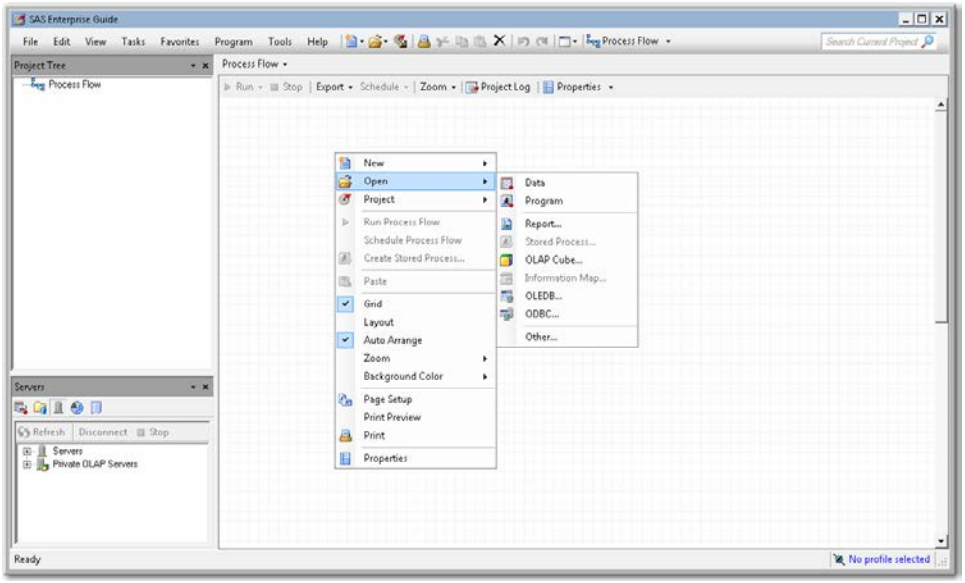


Display 27 - Simple Import Data in SAS Enterprise Guide

The following section delves into the Import Data wizard in SAS Enterprise Guide, the nodes that are generated, and how these nodes are converted when the EGP is opened in SAS Studio.

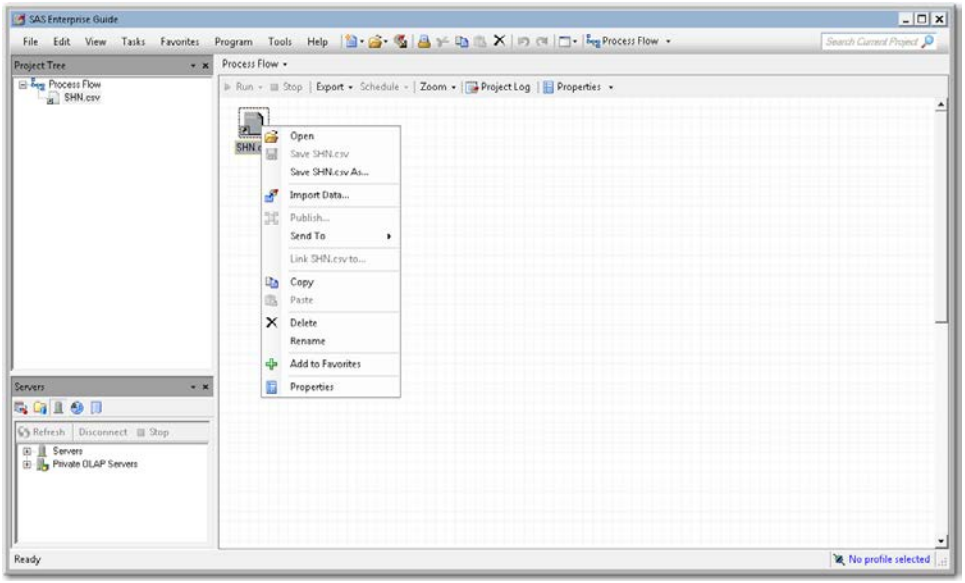
Import Data Wizard

In SAS Enterprise Guide, select **Open** -> **Data** to create a Data node.



Display 28 – Using Open->Data to Create a Data Node

If the Data node is for an external file, you can choose to import the data in that file to a SAS data set. Selecting **Import Data** on the Data node opens the Import Data wizard.



Display 29 - Import Data Option in Pop-Up Menu

The Specify Data panel of the Import Data wizard:

- displays the location of the source data file
- allows you to specify the encoding to use when importing the file
- allows you to specify the output SAS data set where you want to save the imported data

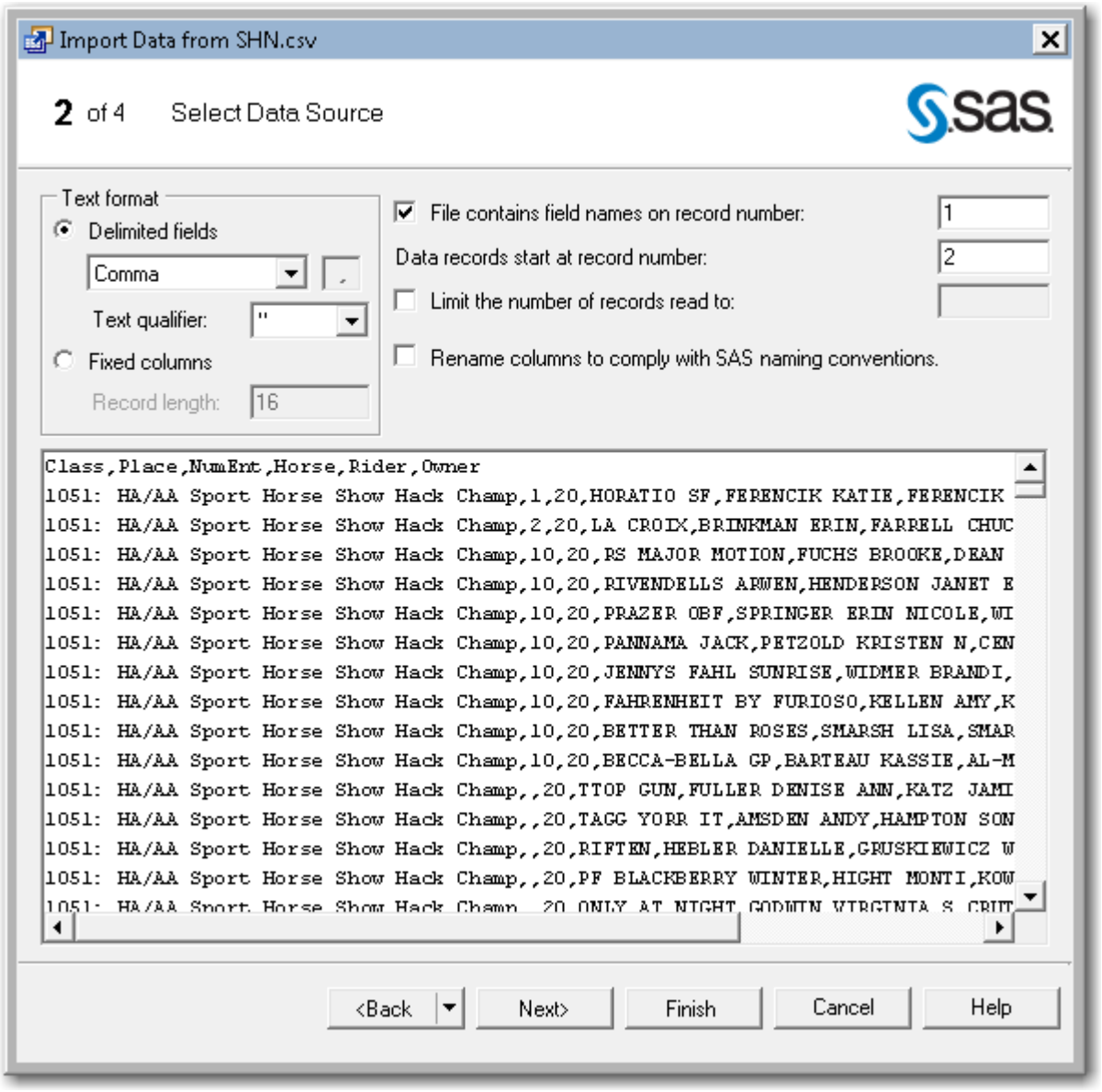
Click **Performance** to control how the wizard processes the data in the next two steps of the wizard.

SAS Enterprise Guide creates a Data node for the output SAS data set. This node is converted to a SAS Program node that displays a view of the SAS data set. The encoding is not yet honored by code generated by SAS Studio.

Display 30 - Specify Data Panel of Import Data Wizard

The Select Data Source panel shows data from the file and allows you to specify options, such as the text format, the record number that contains the field names, the record number where the data starts, and so on.

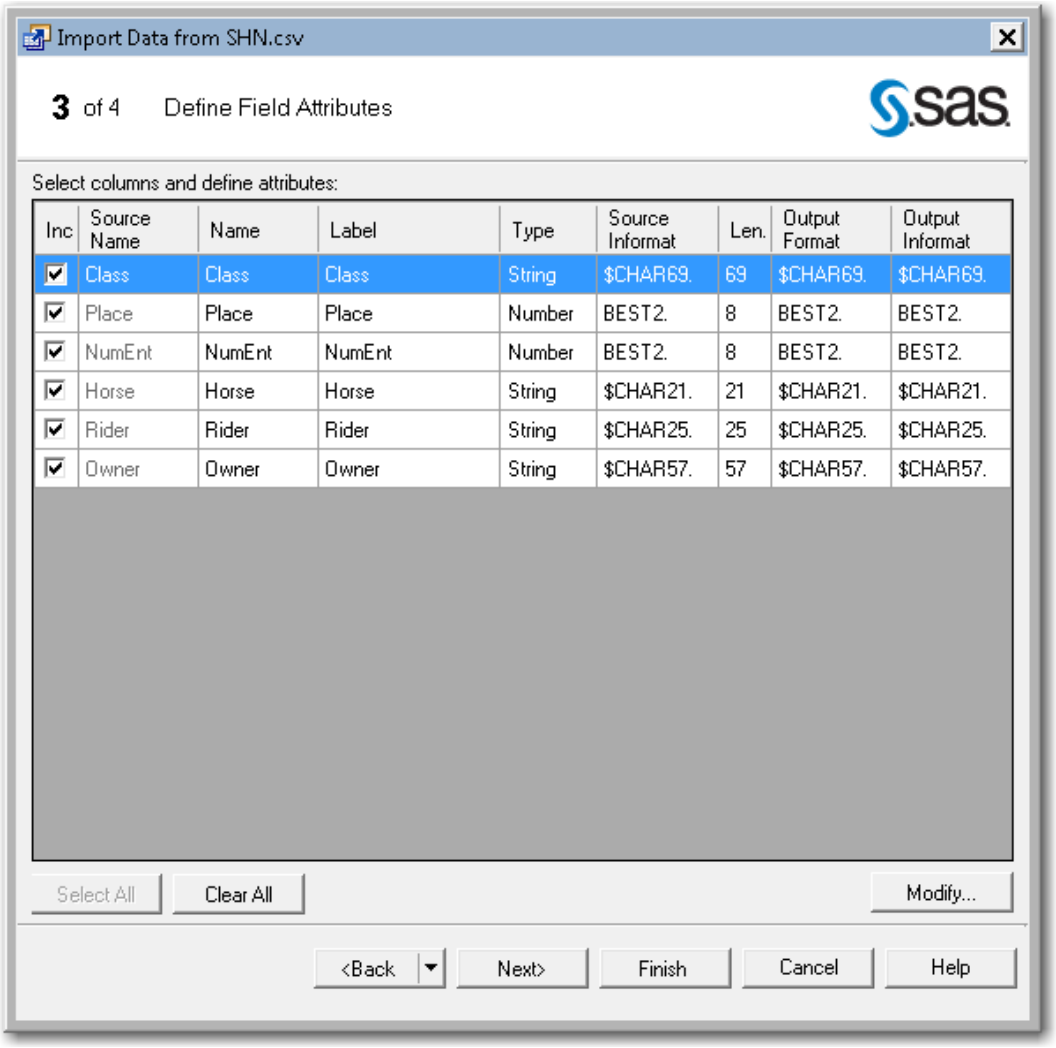
The data collected from this panel is used by SAS Studio when it generates import code.



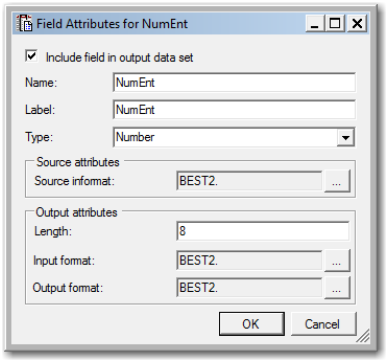
Display 31 - Select Data Source Panel in Import Data Wizard

On the Define Field Attributes panel, you can select which fields you would like to import and specify the formats to use for input and output.

The data collected from this panel is used by SAS Studio when it generates the import code.



Display 31 - Define Field Attributes Panel in Import Data Wizard



Display 32 - Field Attributes Dialog Box for Modifying Field Input and Output Attributes

The Advanced Options panel of the Import Data wizard contains these options:

- **Embed the data within the generated SAS code**

  If this option is checked, the Data node for the input data remains in the EGP and is shown in the converted SAS Studio process flow. However, any changes to the input data are not reflected in the flow in SAS Studio.

- **Import the data using SAS/ACCESS Interface to PC Files whenever possible**

  SAS Studio does not support this functionality.

- **Remove characters that can cause transmission errors from text-based data files**

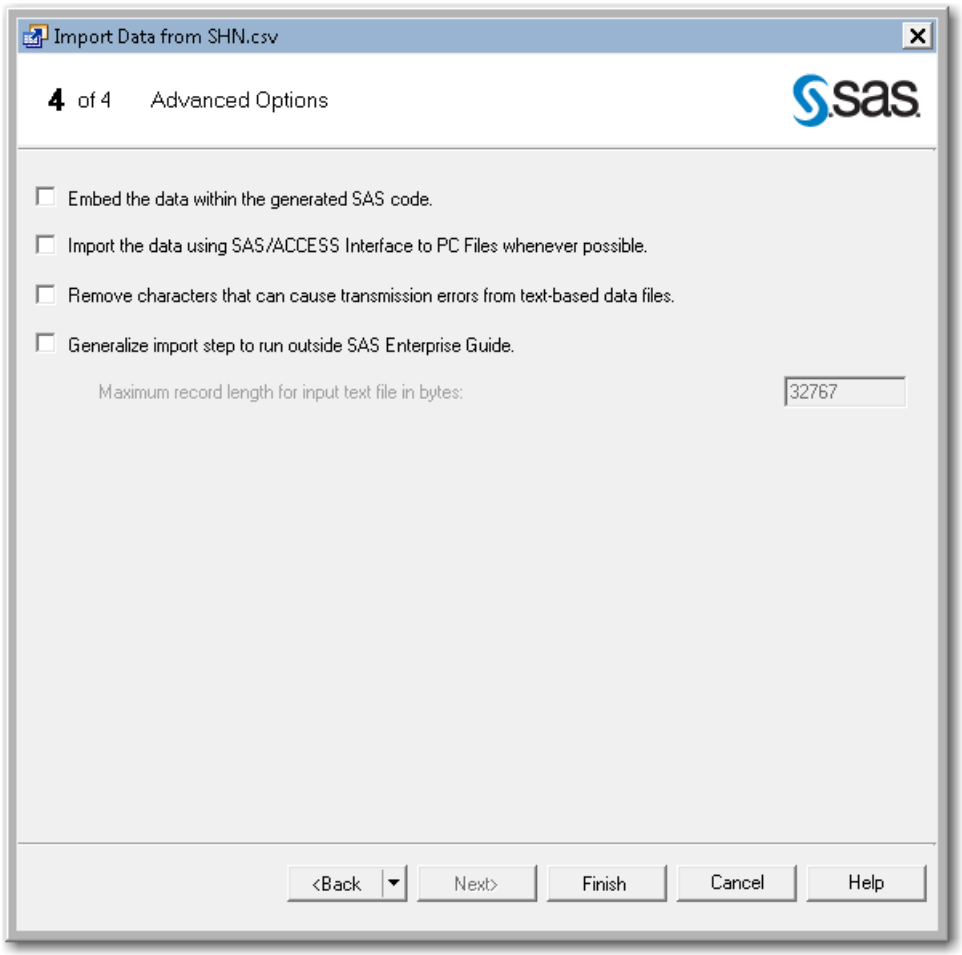
  SAS Studio support for this functionality is unconfirmed.

- **Generalize import step to run outside SAS Enterprise Guide**

  If this option is checked, your converted Import Data node in the SAS Studio process flow will contain the code that was generated by SAS Enterprise Guide for the Import Data node. If this option is not checked, SAS Studio generates the code for the converted Import Code node.

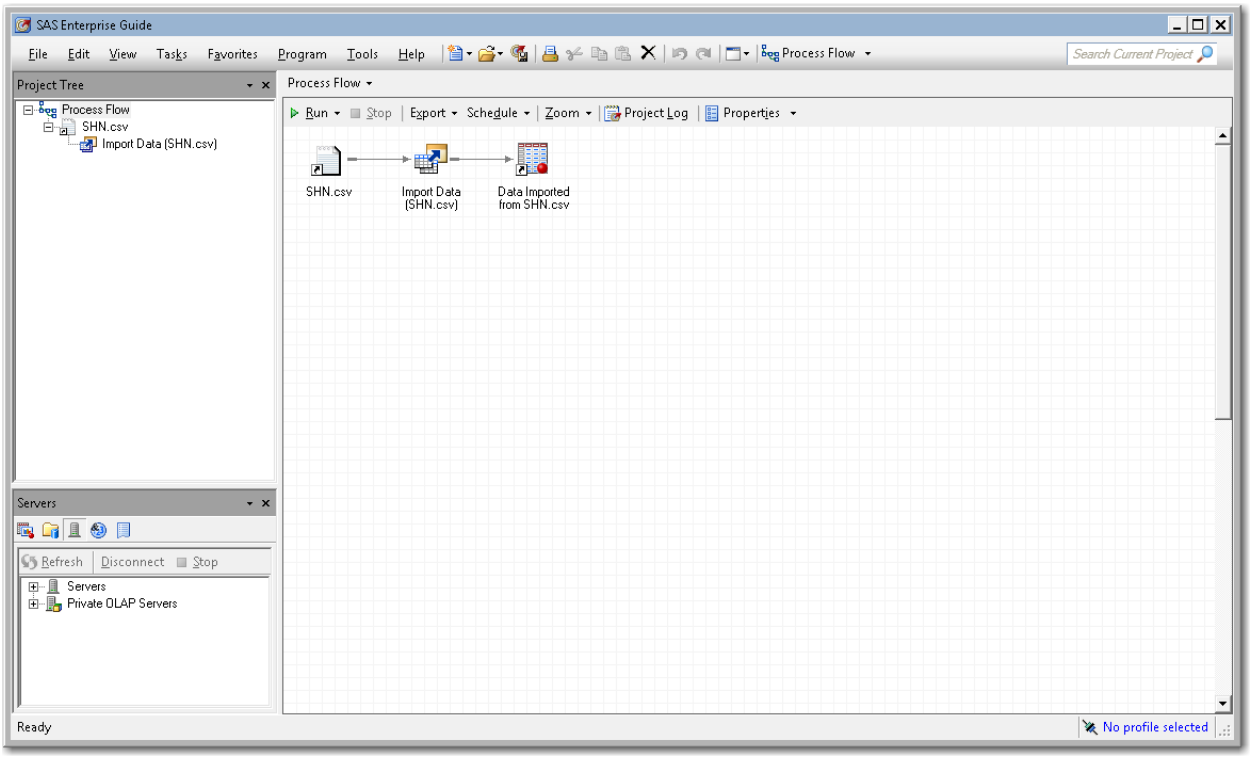
- **Maximum record length for input text file in bytes**

  SAS Studio does not support this functionality.



Display 33 - Advanced Options Panel in the Import Data Wizard

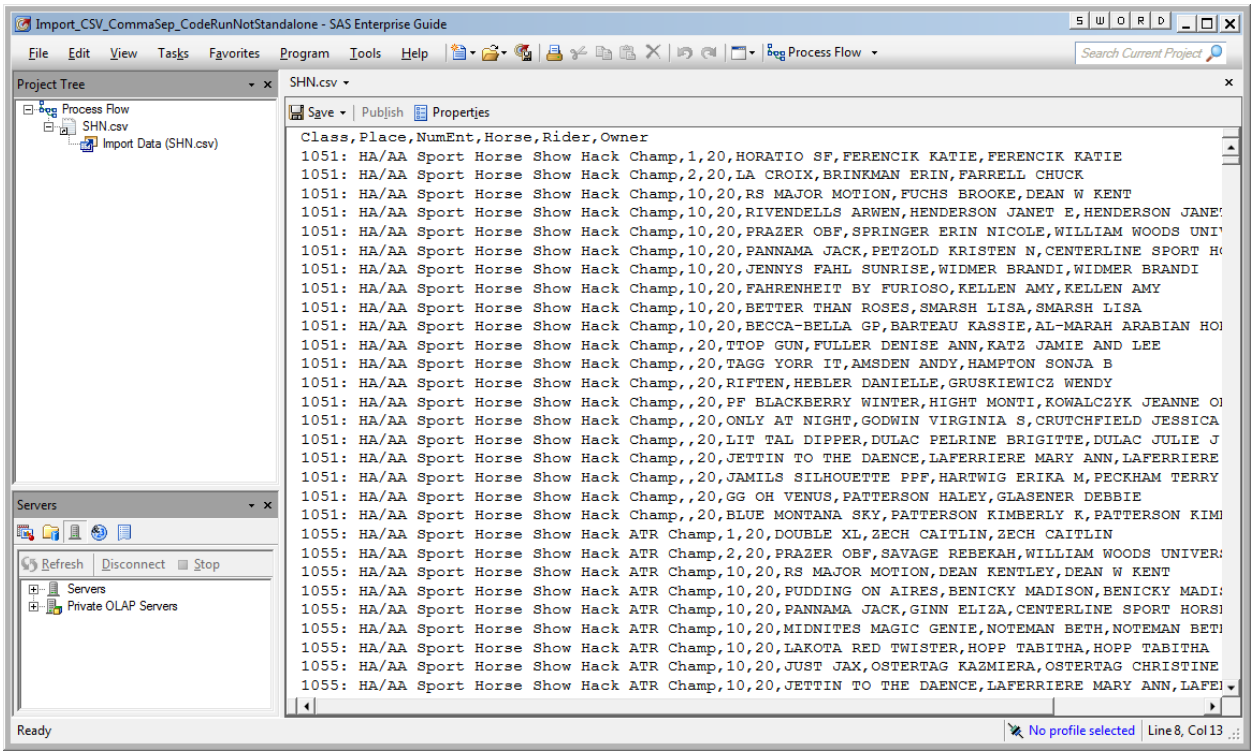
The resulting flow in SAS Enterprise Guide will have a node for the input data, the generated Import Data node, and a node for the the generated output data.



Display 32 - Resulting Process Flow in SAS Enterprise Guide

Data File Node in SAS Enterprise Guide

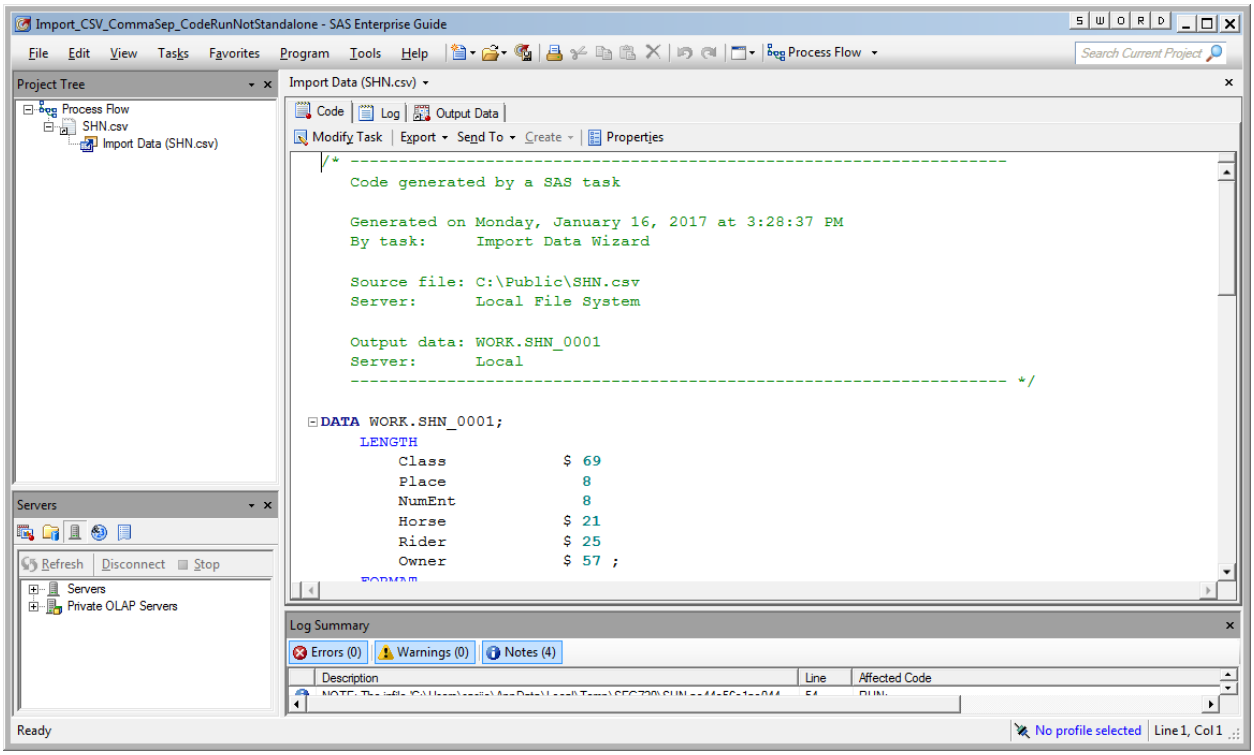
If you open the node for the input data in the process flow, you see the contents of the file to be imported. You can edit the file contents and save your changes. When the EGP is imported to SAS Studio, this type of node becomes a SAS Program node that samples the file contents. You can see an example in Converted Import Data Input File Node.



Display 33 - Contents of Input File for Import Task in SAS Enterprise Guide

Import Data Node in SAS Enterprise Guide

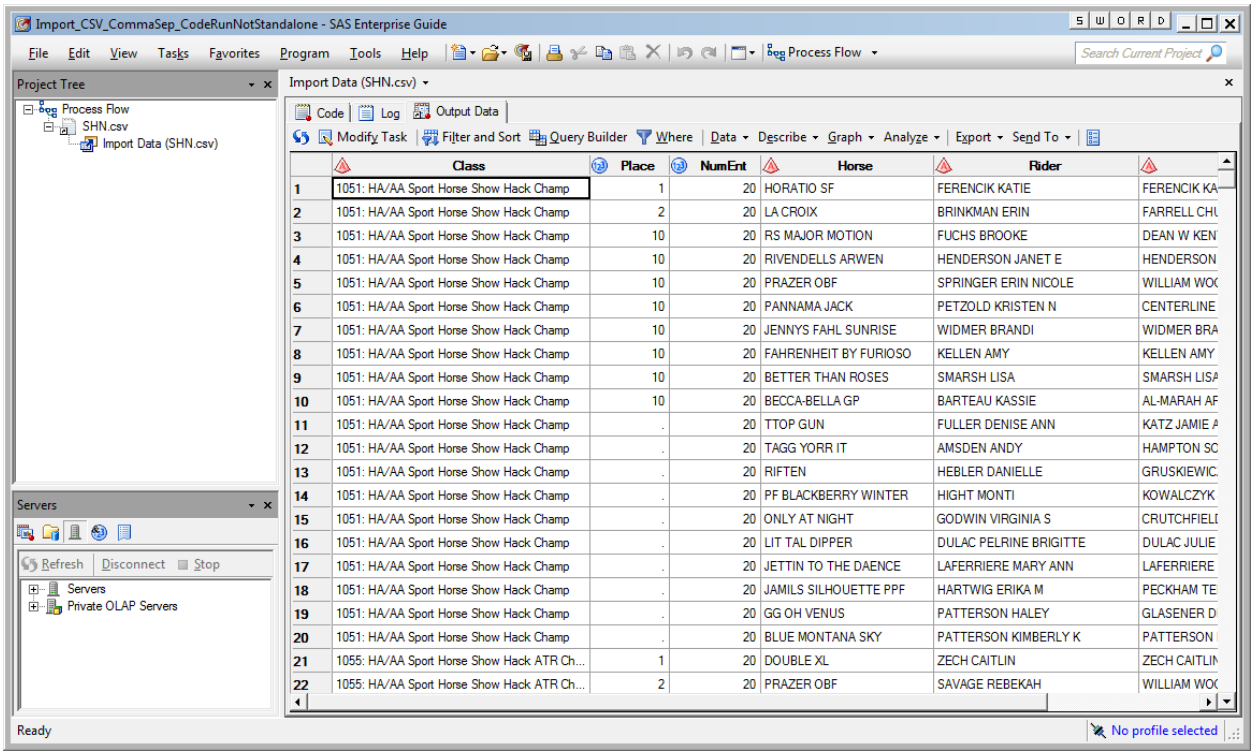
If you open the Import Data node, you see the code generated by SAS Enterprise Guide after running the Specify Data wizard. If you want to change the import options, you cannot edit this code directly. Instead, you must right-click the node and select **Modify Import Data** from the pop-up menu. When the EGP is opened in SAS Studio, the Import Data node is converted to a SAS Program node. You can see an example of a converted SAS Program node in Converted SAS Program Node for Import Data Node.



Display 34 - Import Data Code in SAS Enterprise Guide

SAS Data Set Node in SAS Enterprise Guide

If you open the output data node from the Import Data task, you see a SAS data set that contains the imported data. When the EGP file is opened in SAS Studio, the Data node for the output data set is converted to a SAS Program node that samples the output data set. An example of this can be seen in Converted Output Data Set Node for Import Data Node.

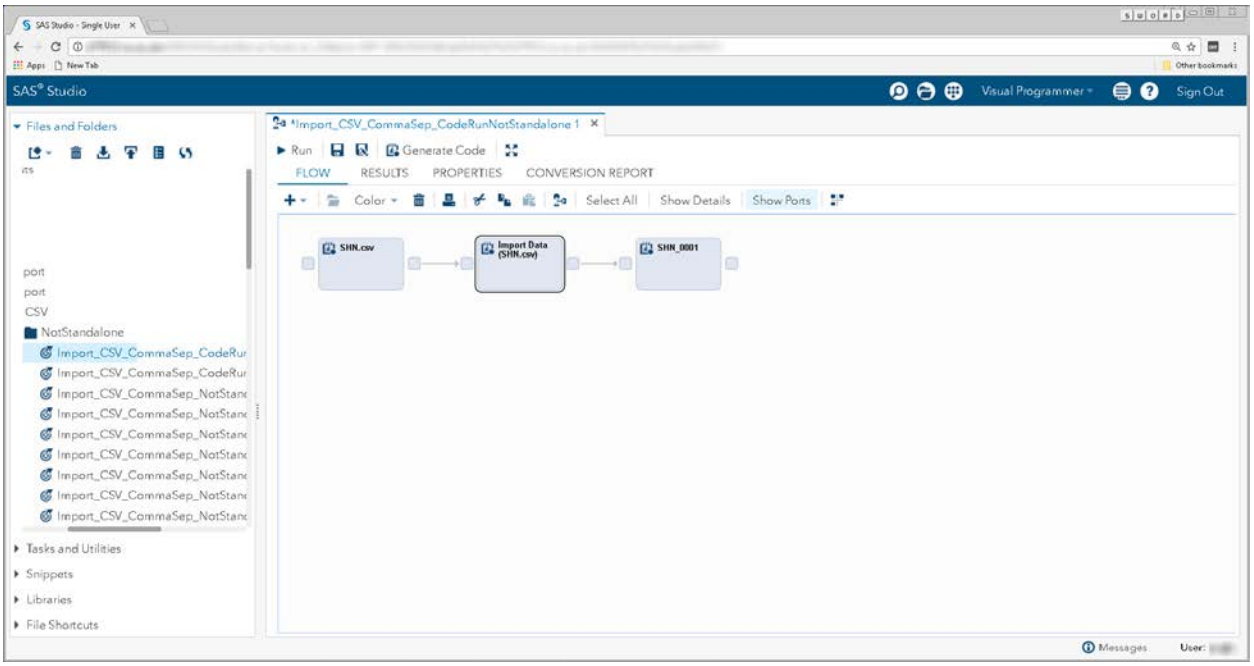


	Class	Place	NumEnt	Horse	Rider	
1	1051: HA/AA Sport Horse Show Hack Champ	1	20	HORATIO SF	FERENCIK KATIE	FERENCIK KA
2	1051: HA/AA Sport Horse Show Hack Champ	2	20	LA CROIX	BRINKMAN ERIN	FARRELL CHU
3	1051: HA/AA Sport Horse Show Hack Champ	10	20	RS MAJOR MOTION	FUCHS BROOKE	DEAN W KEN
4	1051: HA/AA Sport Horse Show Hack Champ	10	20	RIVENDELLS ARWEN	HENDERSON JANET E	HENDERSON
5	1051: HA/AA Sport Horse Show Hack Champ	10	20	PRAZER OBF	SPRINGER ERIN NICOLE	WILLIAM WOO
6	1051: HA/AA Sport Horse Show Hack Champ	10	20	PANNAMA JACK	PETZOLD KRISTEN N	CENTERLINE
7	1051: HA/AA Sport Horse Show Hack Champ	10	20	JENNYS FAHL SUNRISE	WIDMER BRANDI	WIDMER BRA
8	1051: HA/AA Sport Horse Show Hack Champ	10	20	FAHRENHEIT BY FURIOSO	KELLEN AMY	KELLEN AMY
9	1051: HA/AA Sport Horse Show Hack Champ	10	20	BETTER THAN ROSES	SMARSH LISA	SMARSH LISA
10	1051: HA/AA Sport Horse Show Hack Champ	10	20	BECCA-BELLA GP	BARTEAU KASSIE	AL-MARAH AF
11	1051: HA/AA Sport Horse Show Hack Champ	.	20	TTOP GUN	FULLER DENISE ANN	KATZ JAMIE A
12	1051: HA/AA Sport Horse Show Hack Champ	.	20	TAGG YORR IT	AMSDEN ANDY	HAMPTON SC
13	1051: HA/AA Sport Horse Show Hack Champ	.	20	RIFTEN	HEBLER DANIELLE	GRUSKIEWIC
14	1051: HA/AA Sport Horse Show Hack Champ	.	20	PF BLACKBERRY WINTER	HIGHT MONTI	KOWALCZYK
15	1051: HA/AA Sport Horse Show Hack Champ	.	20	ONLY AT NIGHT	GODWIN VIRGINIA S	CRUTCHFIELI
16	1051: HA/AA Sport Horse Show Hack Champ	.	20	LIT TAL DIPPER	DULAC PELRINE BRIGITTE	DULAC JULIE
17	1051: HA/AA Sport Horse Show Hack Champ	.	20	JETTIN TO THE DAENCE	LAFERRIERE MARY ANN	LAFERRIERE
18	1051: HA/AA Sport Horse Show Hack Champ	.	20	JAMILS SILHOUETTE PPF	HARTWIG ERIKA M	PECKHAM TE
19	1051: HA/AA Sport Horse Show Hack Champ	.	20	GG OH VENUS	PATTERSON HALEY	GLASENER D
20	1051: HA/AA Sport Horse Show Hack Champ	.	20	BLUE MONTANA SKY	PATTERSON KIMBERLY K	PATTERSON
21	1055: HA/AA Sport Horse Show Hack ATR Ch...	1	20	DOUBLE XL	ZECH CAITLIN	ZECH CAITLIN
22	1055: HA/AA Sport Horse Show Hack ATR Ch...	2	20	PRAZER OBF	SAVAGE REBEKAH	WILLIAM WOO

Display 35 - Output Data Set in SAS Enterprise Guide

SAS Studio

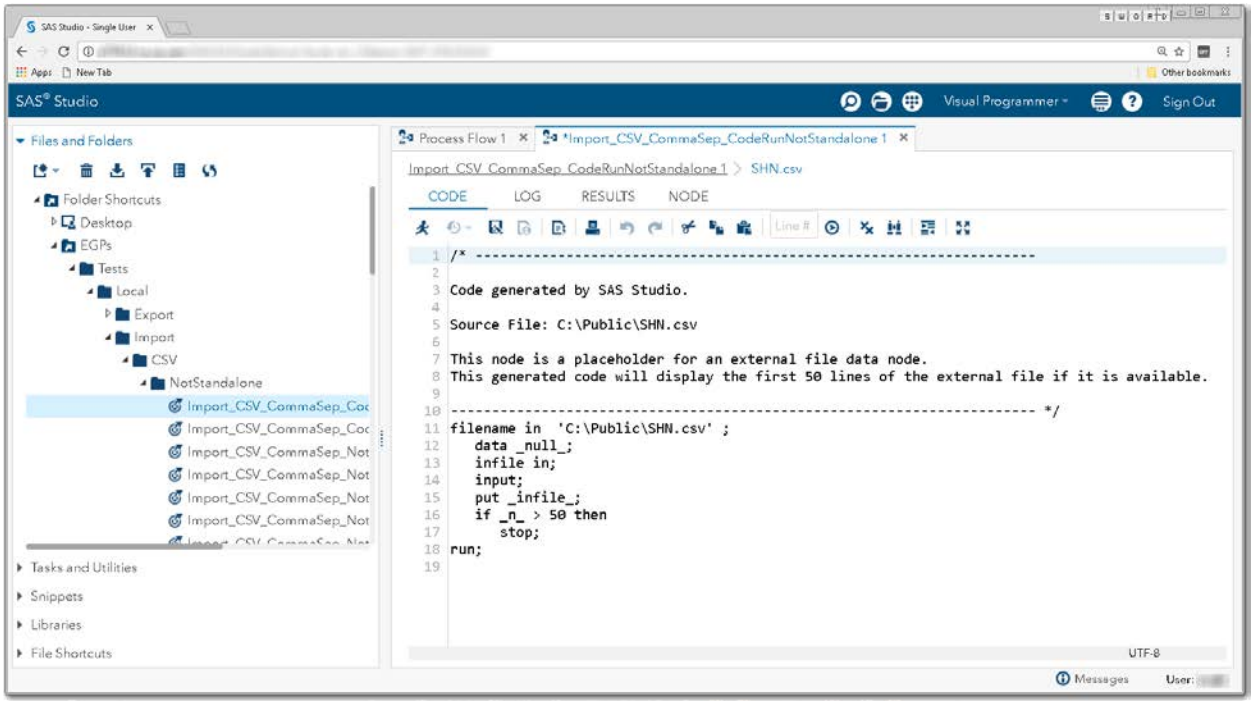
Opening the EGP in SAS Studio results in representative nodes for each node in the Import Data process flow, but these nodes are remarkably different in SAS Studio.



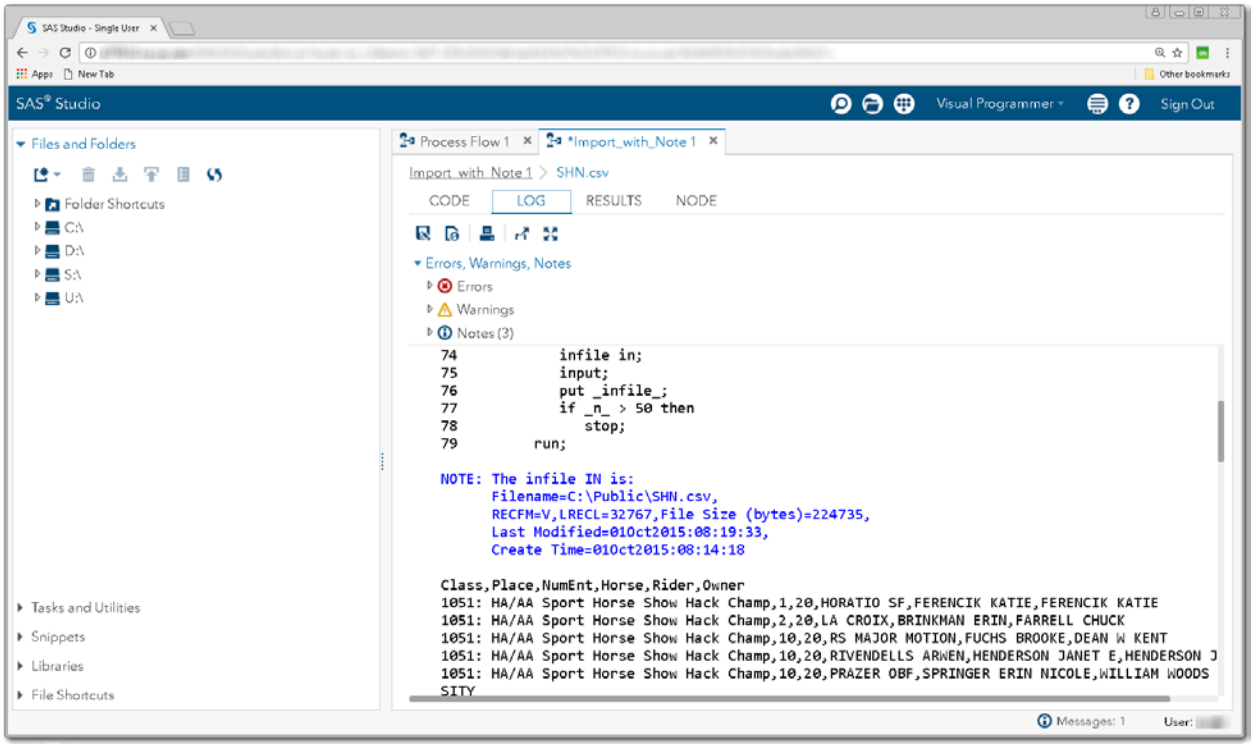
Display 36 - Converted Process Flow for Import Data Task in SAS Studio

Converted Import Data Input File Node

The input Data node for the Import Data node is converted to a SAS Program node in SAS Studio. The SAS Program node contains SAS Studio generated code that samples the input file contents. See the Data Sampling section of this document for more information about file sampling Data nodes. You can contrast this to the Data File Node in SAS Enterprise Guide.



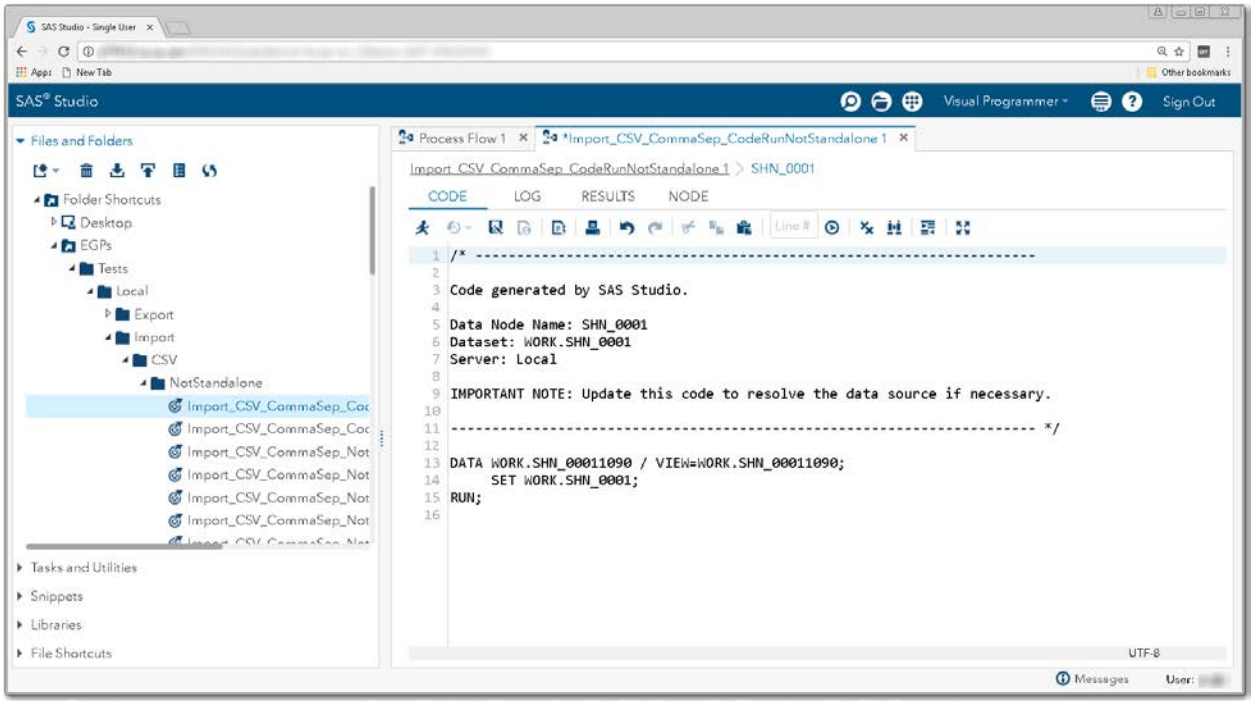
Display 37 - Converted Input Data File Node in SAS Studio



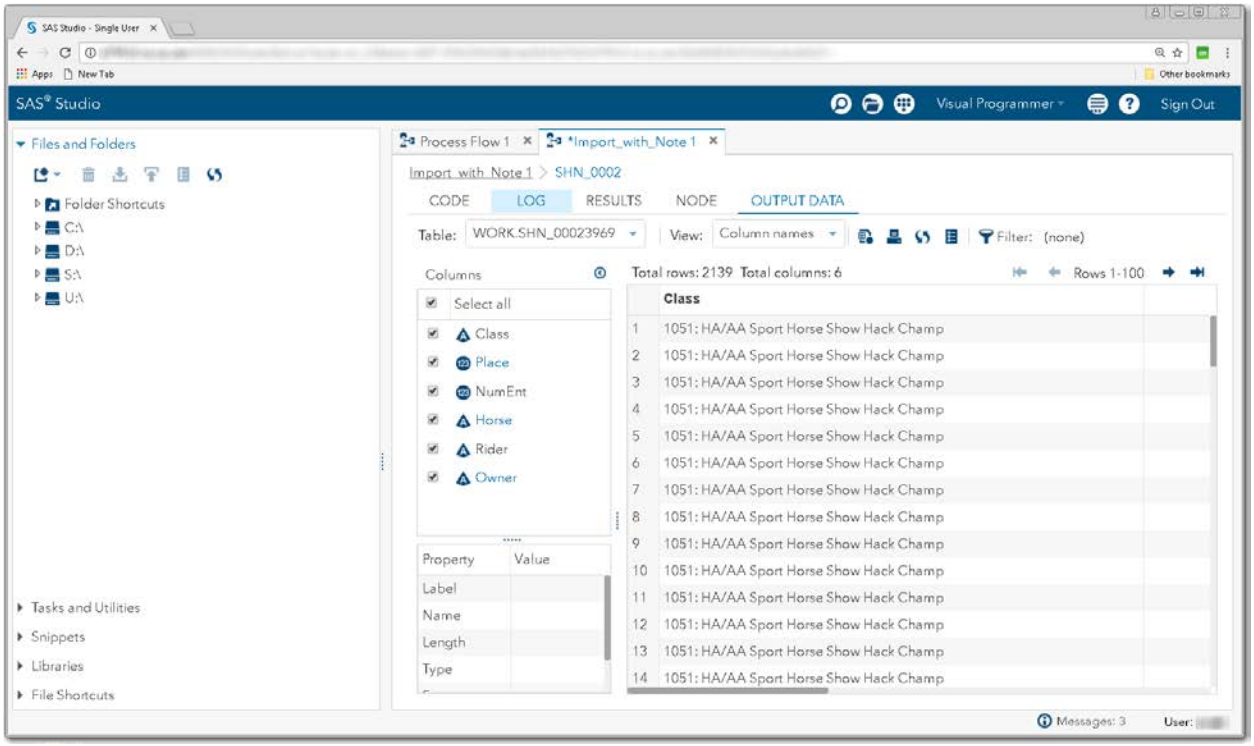
Display 38 – Log for Converted Data Node

Converted Output Data Set Node for Import Data Node

Similiarly, the node for the output data has been converted to a SAS Program node. When you run the converted process flow, the **OUTPUT DATA** tab for the converted node contains a view of the imported SAS data set.



Display 39 – Contents of SAS Program Node for Output Data

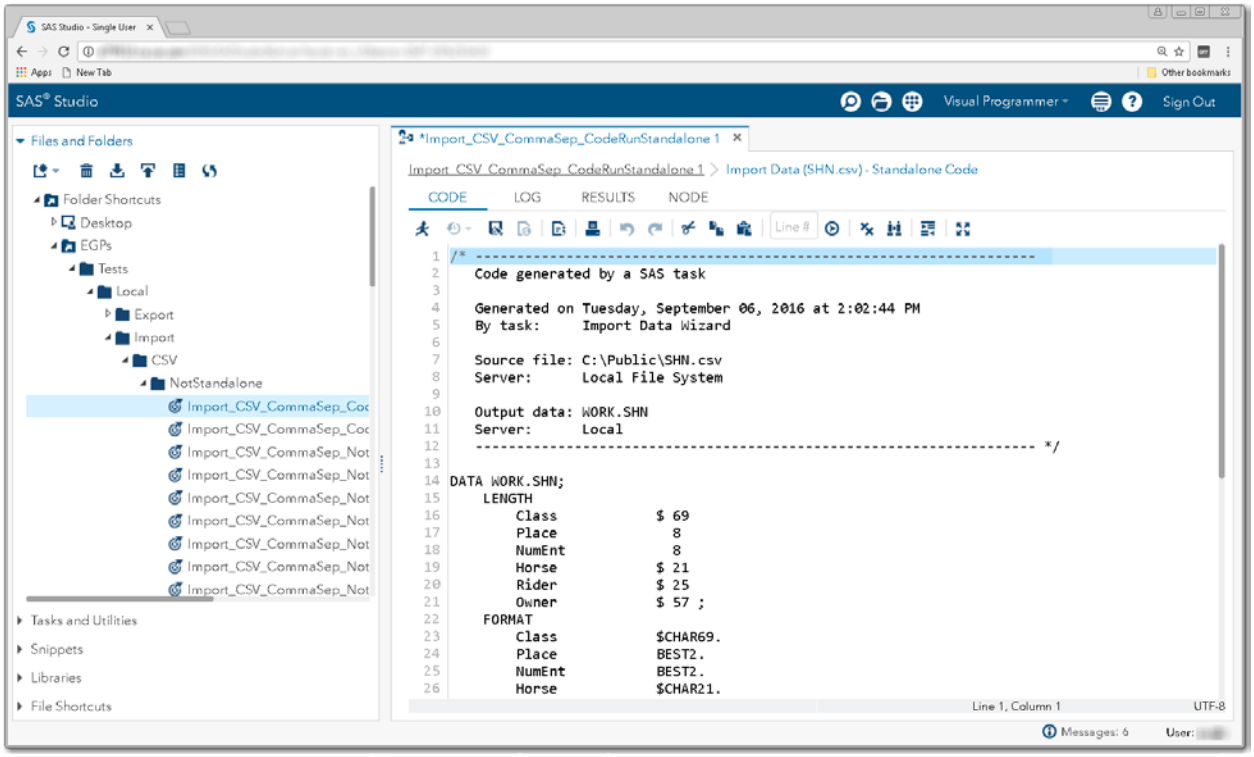


Display 40 – Output Data for Converted Node

Converted SAS Program Node for Import Data Node

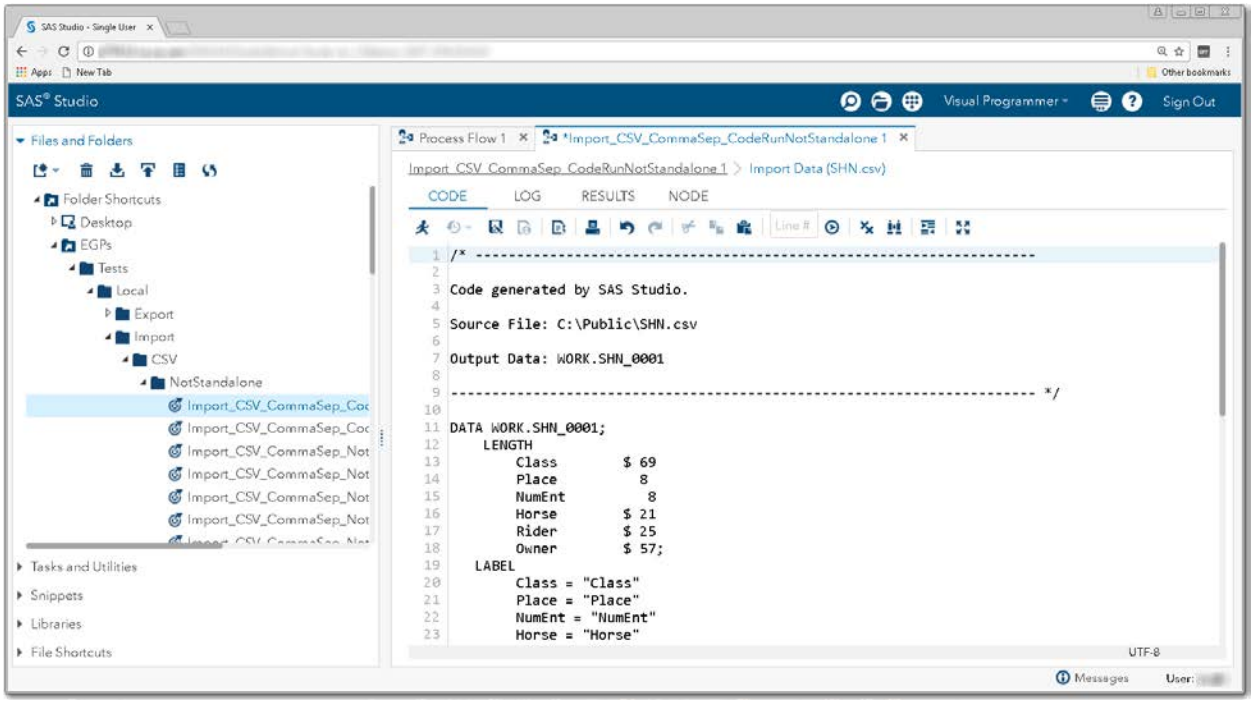
An Import Data node in SAS Enterprise Guide becomes a SAS Program node in SAS Studio. You can contrast this to the Import Data Node in SAS Enterprise Guide. Unlike the Import Data node in SAS Enterprise Guide, the code in the converted SAS Program node can be edited. The contents of this node will be **either**:

- code generated by SAS Enterprise Guide if the **Generalize step to run outside of SAS Enterprise Guide** option was selected in the Specify Data panel in the Import Task wizard. This code should function the same in SAS Studio and SAS Enterprise Guide when using an identical SAS server.



Display 41 – Code Generated in SAS Enterprise Guide Is the Same in SAS Studio

- code generated by SAS Studio that tries to closely match the functionality provided by Import Data in SAS Enterprise Guide.



Display 42 – Code Generated by SAS Studio for Converted Import Data Node

Unsupported Import Features and Limitations

Encoding

Encoding support is on the list of features to be added.

Import code generated by SAS Studio uses the default file encoding when reading the input data. If the default encoding is not applicable to your input file, add the INFILE statement to the code generated by SAS Studio to manually set the encoding of your input data.

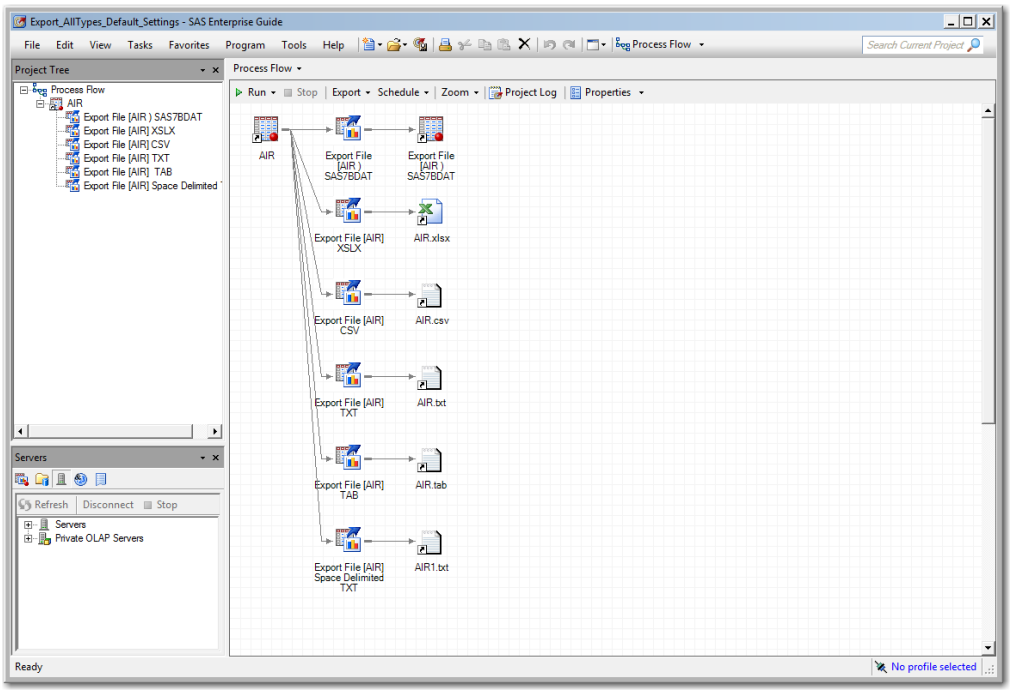
```
INFILE 'C:\Public\SHN.csv'
       LRECL=32767
       FIRSTOBS=2
       ENCODING="WLATIN1"
       DLM='2c'x
       MISSOVER
       DSD ;
```

Unsupported Import Data

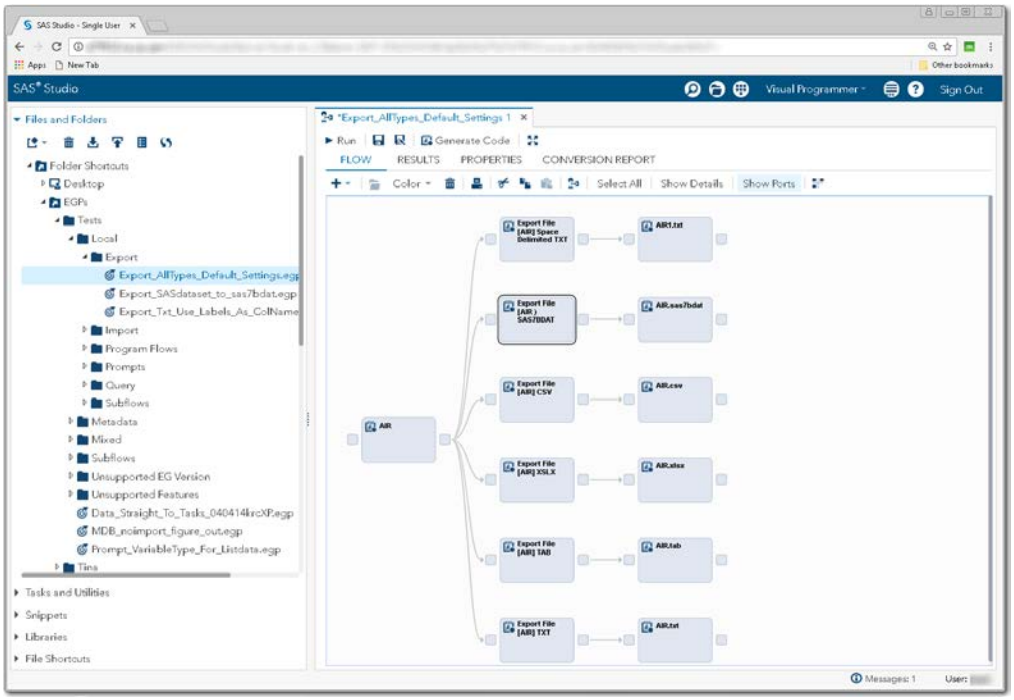
SAS Studio does not support importing data from HTML files. Currently, the code generated for importing HTML content is nonsensical.

EXPORT FILE NODES

Because an Export File node in SAS Enterprise Guide does not port directly to a SAS Studio node, SAS Studio makes quite a few changes to Export File nodes. While the nodes associated with exporting files in SAS Enterprise Guide are all represented in a converted process flow, the nodes are remarkably different. The following two displays show the same Export File process flow in SAS Enterprise Guide and SAS Studio.



Display 43 – Export File Nodes in SAS Enterprise Guide



Display 44 - Converted Export File Nodes in SAS Studio

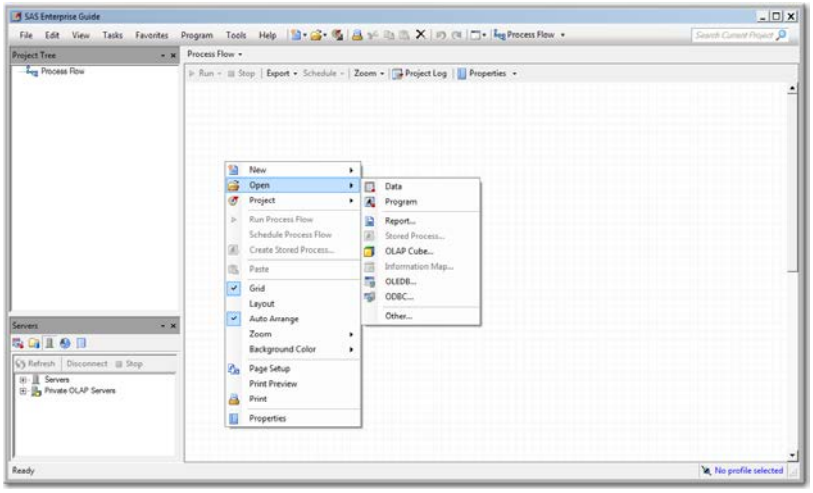
Enterprise Guide

Exporting Files in SAS Enterprise Guide

In the following example, a SAS data set is added to the SAS Enterprise Guide project and the Export <Data Set> as a Step in the Project option is used to specify the output data location, delimiter information, fields to import, and other advanced options. The resulting process flow contains a Data node for the input data set, an Export File node, and a Data node for the output data.

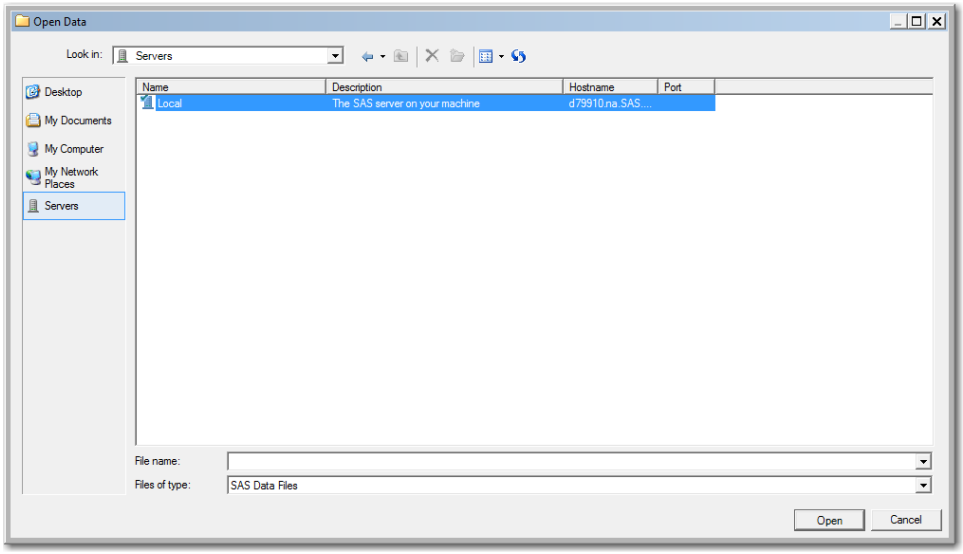
Select SAS Data Set to Export

1. Right-click in the project and select **Open** -> **Data**.



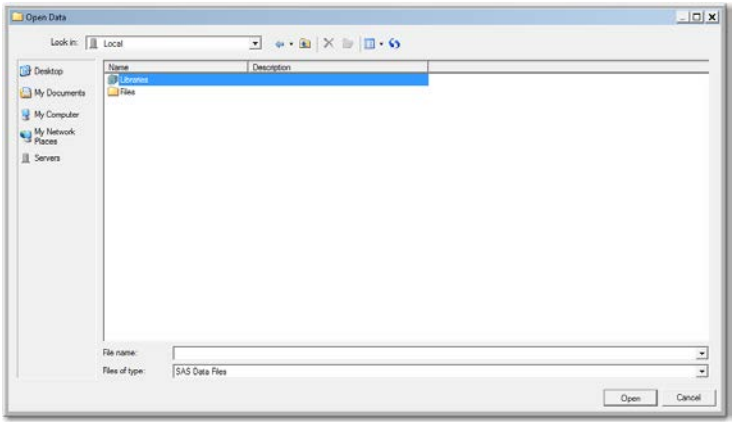
Display 45 – Opening Your Data from the Process Flow

2. Select the server that contains the SAS data set you want to export.



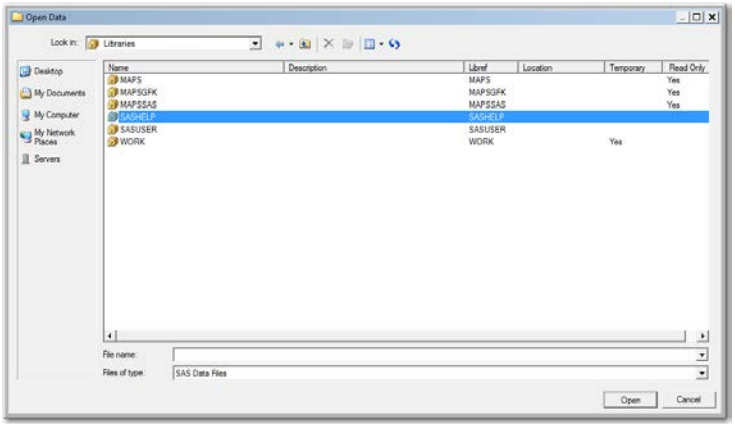
Display 46 – Selecting a Server in the Open Data Dialog Box

3. Select **Libraries**.



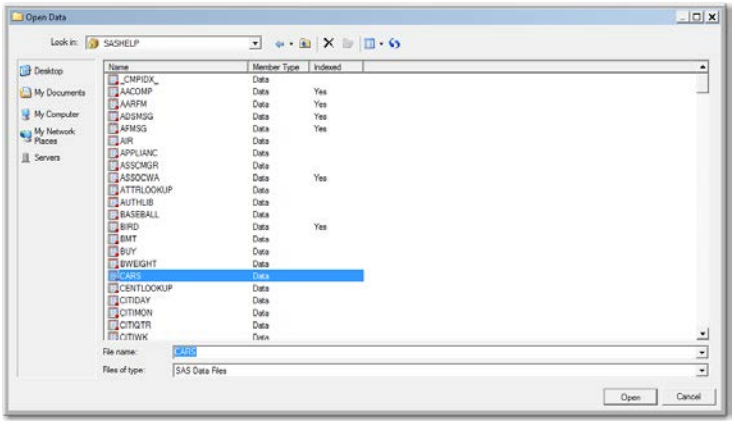
Display 47 – Opening the Libraries Folder in the Open Data Dialog Box

4. Select the library that contains the SAS data set you want to export.



Display 48 – Select the Library That Contains the SAS Data Set

5. Select the SAS data set you want to export.

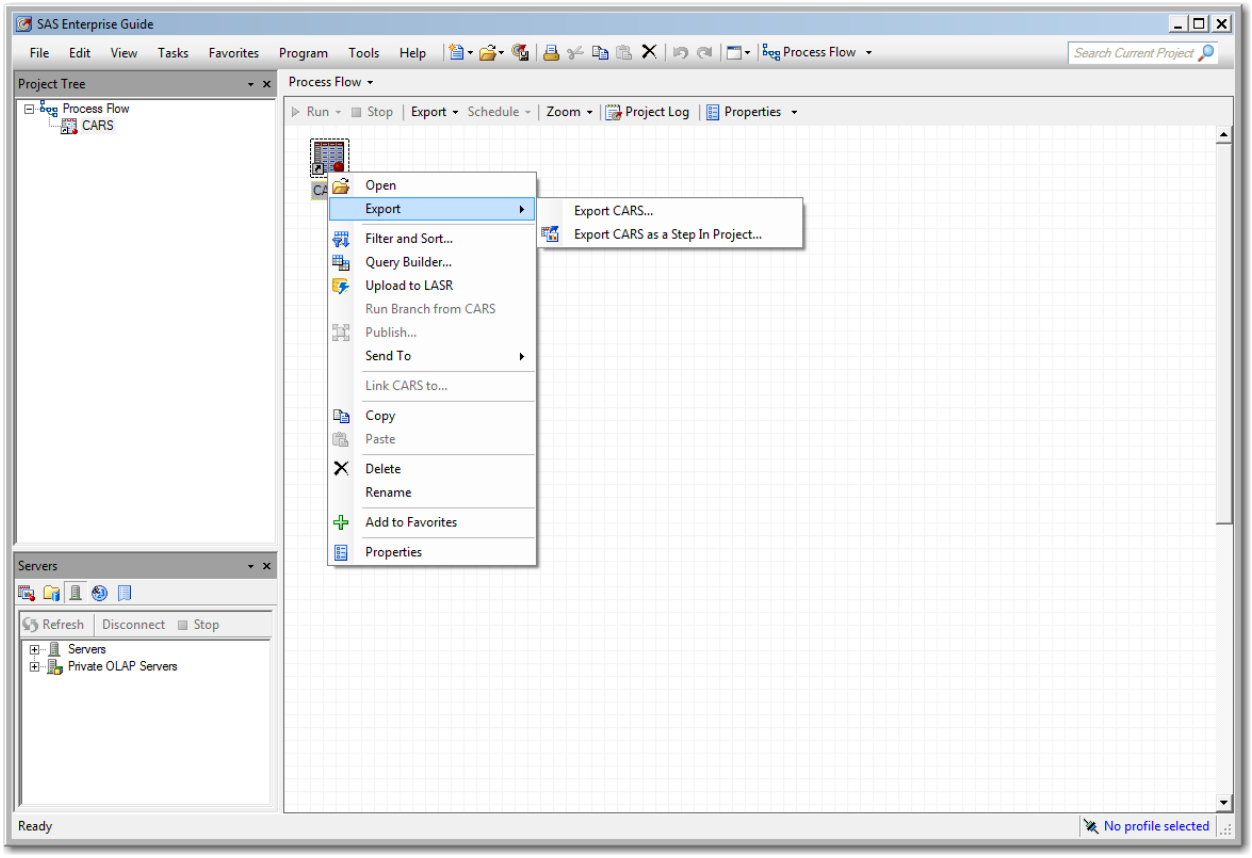


Display 49 – Select the Data Set in the Open Data Dialog Box

A Data node for the SAS data set appears in your SAS Enterprise Guide project.

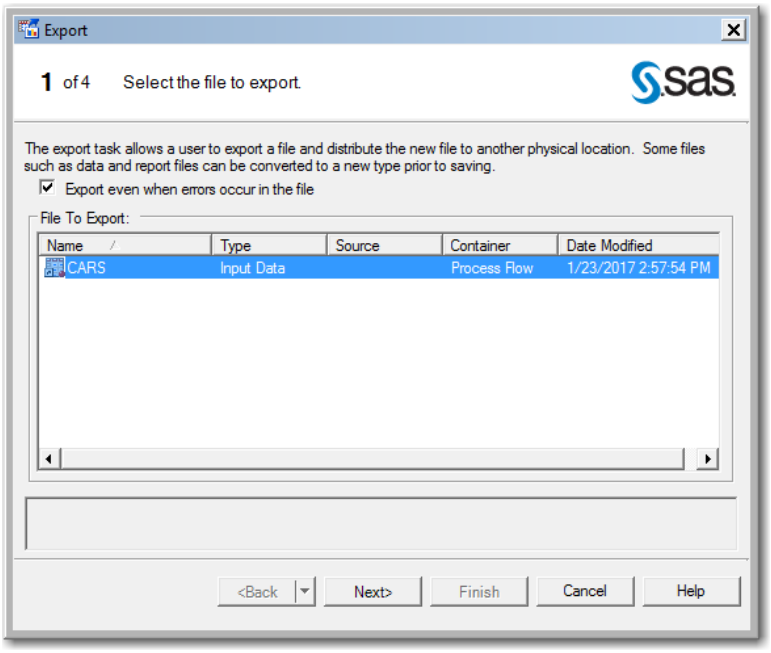
Export <Data Set> as a Step in Project

Right-click the Data node for the SAS data set to use the **Export <Data Set> as a Step In Project** menu item to open the Export wizard**.**



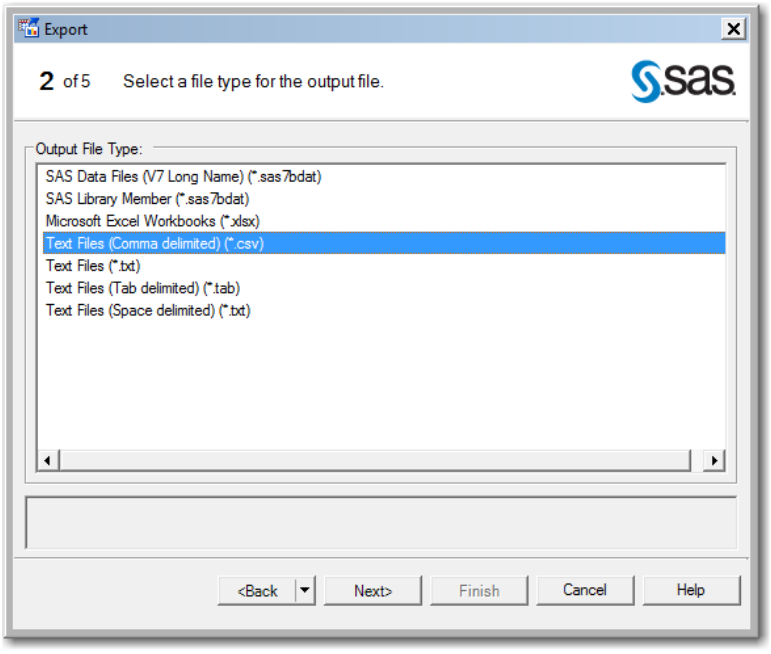
Display 50 – Exporting a Data Node in SAS Enterprise Guide

The first panel of the Export wizard displays the SAS data set you have chosen to export. There is an **Export even when errors occur in the file** check box in this panel. The impact of this check box on a converted EGP in SAS Studio has not been researched.



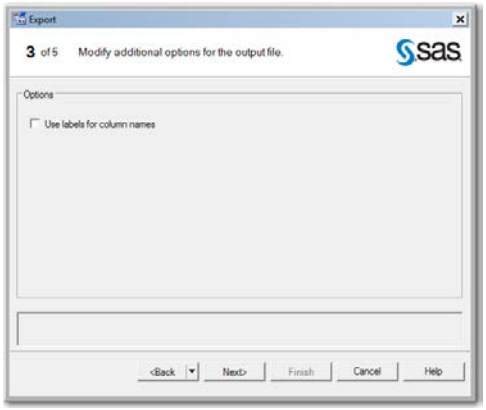
Display 51 – Selecting the File to Export in the Export Wizard

In the second step of the Export wizard, specify the type of the output file for the exported SAS data set. SAS Studio supports all file types.



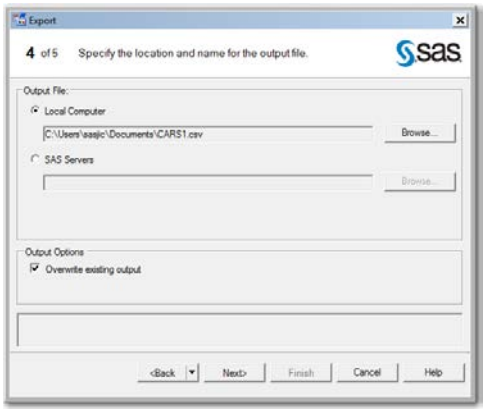
Display 52 - Selecting the File Type for the Exported Data

The third step has the **Use labels for column names** check box. SAS Studio supports this option.



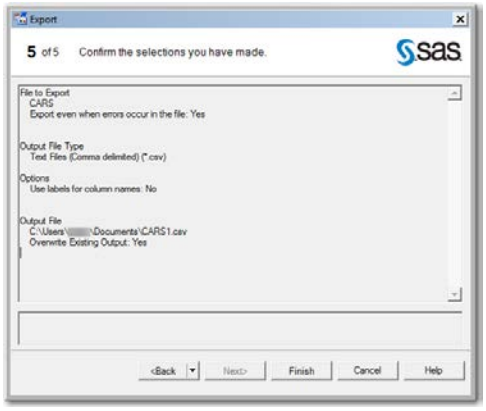
Display 53 - Additional Options in the Export Wizard

In the fourth step of the Export wizard, you specify where to write the exported output file. SAS Studio can write this file only to the file system available to its connected workspace server. If the workspace server in SAS Enterprise Guide is different from what is specified in SAS Studio, the Conversion Report will display a mismatched environment message.



Display 54 - Specifying the Output Location in the Export Wizard

The final step provides a summary of the selections you have made.

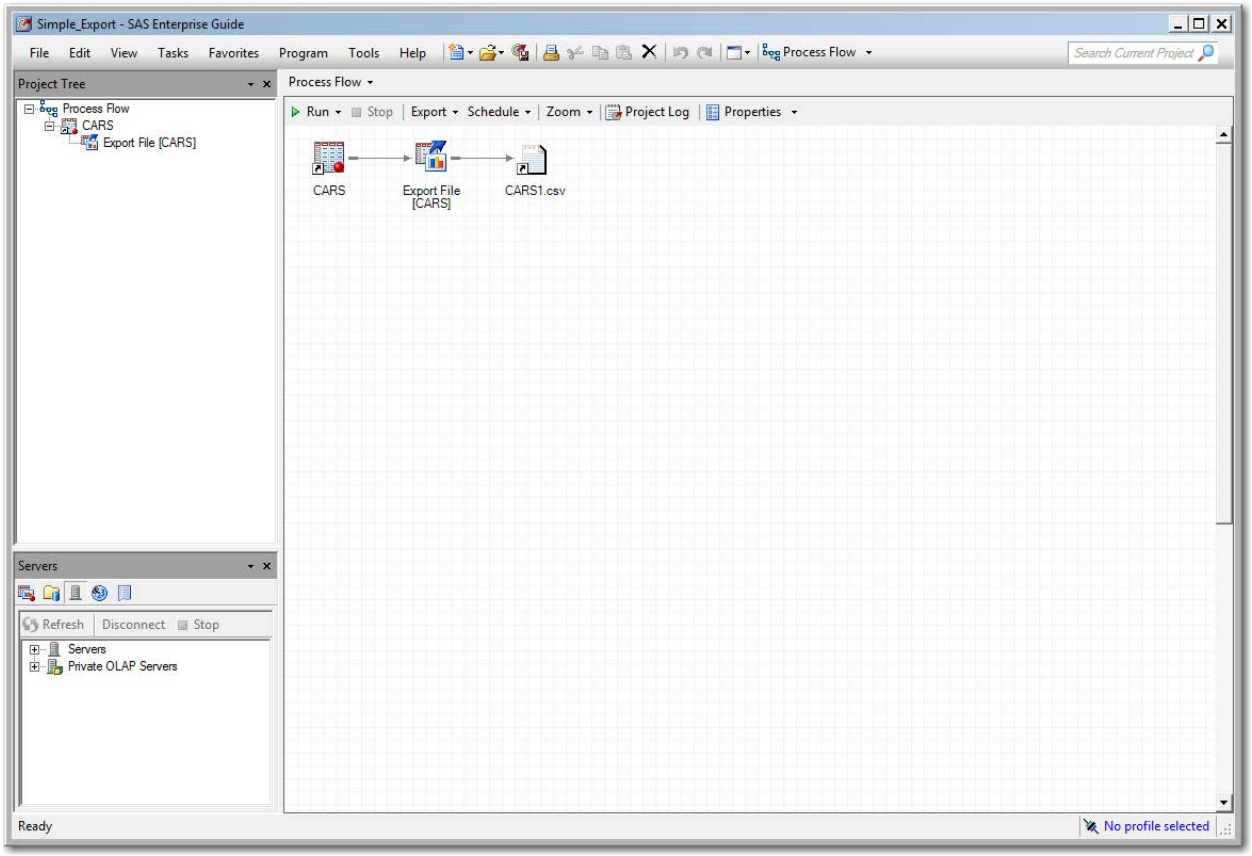


Display 55 - Summary Panel in the Export Wizard

The following display shows the resulting process flow when you run the Export wizard.

Process Flow for Export File in SAS Enterprise Guide

The process flow for Export File contains a Data node for the input SAS data set, an Export File node, and a Data node for the output file. You can contrast this flow to the Converted Export Flow in SAS Studio.

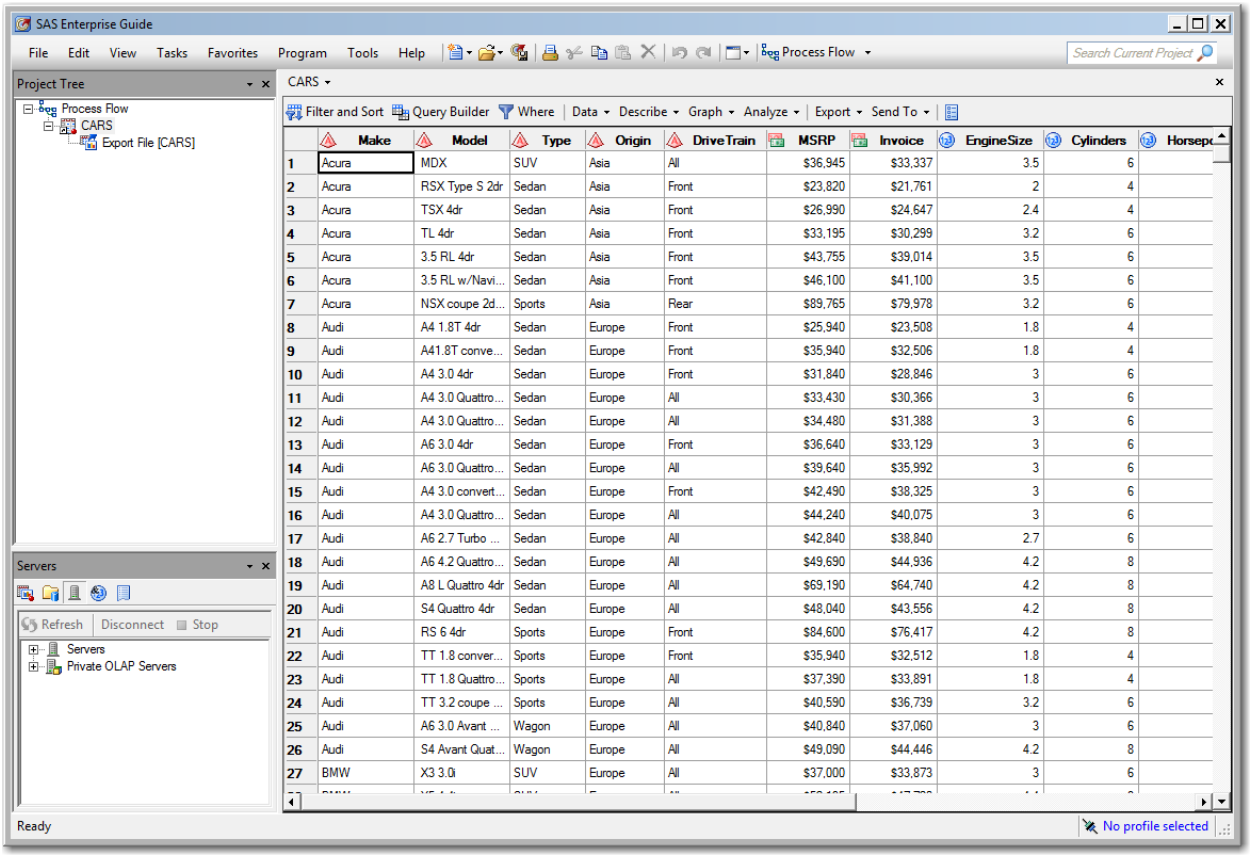


Display 56 - Resulting Process Flow in SAS Enterprise Guide

Nodes in the Process Flow for Export File in SAS Enterprise Guide

Node for Input SAS Data Set in Enterprise Guide

The first node in the process flow is a Data node for the input SAS data set. Open this node to see the contents of the data set, update the data, as well as many other functions. You can contrast this functionality to the converted Data Set Node for Input Data in SAS Studio.

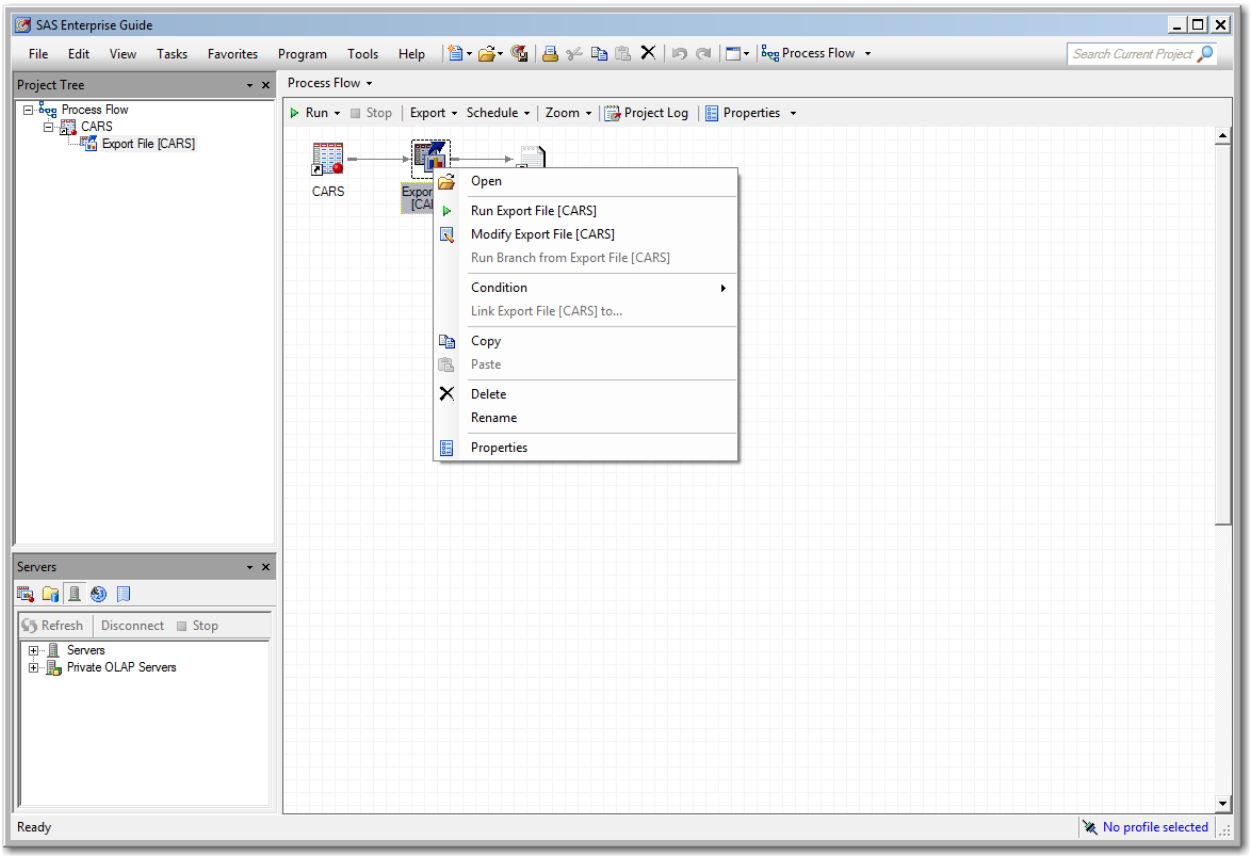


	Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize	Cylinders	Horsep
1	Acura	MDX	SUV	Asia	All	$36,945	$33,337	3.5	6	
2	Acura	RSX Type S 2dr	Sedan	Asia	Front	$23,820	$21,761	2	4	
3	Acura	TSX 4dr	Sedan	Asia	Front	$26,990	$24,647	2.4	4	
4	Acura	TL 4dr	Sedan	Asia	Front	$33,195	$30,299	3.2	6	
5	Acura	3.5 RL 4dr	Sedan	Asia	Front	$43,755	$39,014	3.5	6	
6	Acura	3.5 RL w/Navi...	Sedan	Asia	Front	$46,100	$41,100	3.5	6	
7	Acura	NSX coupe 2d...	Sports	Asia	Rear	$89,765	$79,978	3.2	6	
8	Audi	A4 1.8T 4dr	Sedan	Europe	Front	$25,940	$23,508	1.8	4	
9	Audi	A41.8T conve...	Sedan	Europe	Front	$35,940	$32,506	1.8	4	
10	Audi	A4 3.0 4dr	Sedan	Europe	Front	$31,840	$28,846	3	6	
11	Audi	A4 3.0 Quattro...	Sedan	Europe	All	$33,430	$30,366	3	6	
12	Audi	A4 3.0 Quattro...	Sedan	Europe	All	$34,480	$31,388	3	6	
13	Audi	A6 3.0 4dr	Sedan	Europe	Front	$36,640	$33,129	3	6	
14	Audi	A6 3.0 Quattro...	Sedan	Europe	All	$39,640	$35,992	3	6	
15	Audi	A4 3.0 convert...	Sedan	Europe	Front	$42,490	$38,325	3	6	
16	Audi	A4 3.0 Quattro...	Sedan	Europe	All	$44,240	$40,075	3	6	
17	Audi	A6 2.7 Turbo ...	Sedan	Europe	All	$42,840	$38,840	2.7	6	
18	Audi	A6 4.2 Quattro...	Sedan	Europe	All	$49,690	$44,936	4.2	8	
19	Audi	A8 L Quattro 4dr	Sedan	Europe	All	$69,190	$64,740	4.2	8	
20	Audi	S4 Quattro 4dr	Sedan	Europe	All	$48,040	$43,556	4.2	8	
21	Audi	RS 6 4dr	Sports	Europe	Front	$84,600	$76,417	4.2	8	
22	Audi	TT 1.8 conver...	Sports	Europe	Front	$35,940	$32,512	1.8	4	
23	Audi	TT 1.8 Quattro...	Sports	Europe	All	$37,390	$33,891	1.8	4	
24	Audi	TT 3.2 coupe ...	Sports	Europe	All	$40,590	$36,739	3.2	6	
25	Audi	A6 3.0 Avant ...	Wagon	Europe	All	$40,840	$37,060	3	6	
26	Audi	S4 Avant Quat...	Wagon	Europe	All	$49,090	$44,446	4.2	8	
27	BMW	X3 3.0i	SUV	Europe	All	$37,000	$33,873	3	6	

Display 57 - Contents of Data Node for Input SAS Data Set in SAS Enterprise Guide

Export File Node in SAS Enterprise Guide

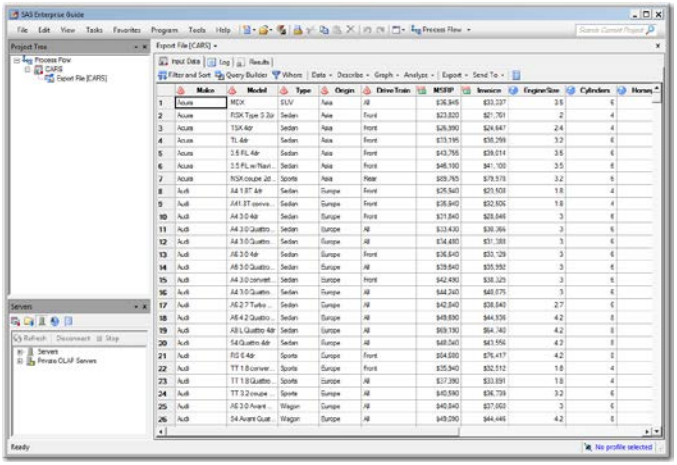
Right-click the Export File node to open or modify the export file. If you select **Modify Export File**, the Export wizard opens, so you can change your export settings. You can contrast this node to the Converted Export File Node in SAS Studio.



Display 58 - Export File Node in SAS Enterprise Guide

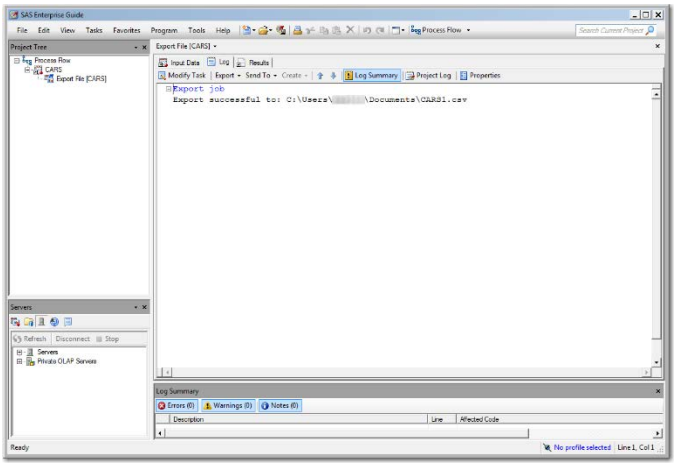
If you select **Open** from the pop-up menu**,** you see the **Input Data**, **Log**, and **Results** tabs as shown in the following displays.

The **Input Data** tab shows the contents of the input SAS data set.



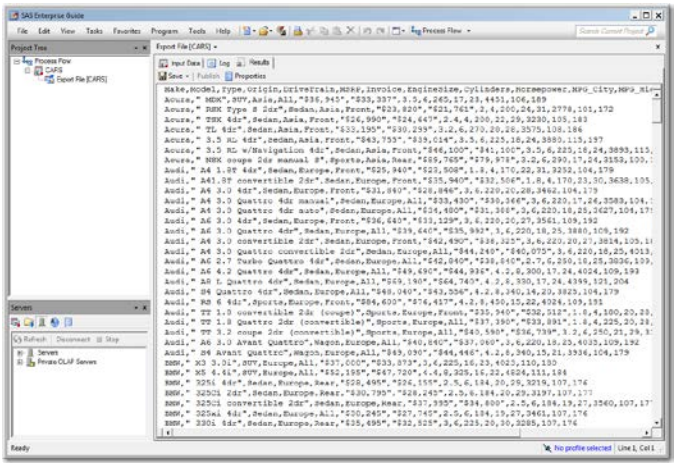
Display 59 - Contents of Input Data Tab for Export File Node in SAS Enterprise Guide

The **Log** tab provides the export status.



Display 60 - Log Tab for Export File Node in SAS Enterprise Guide

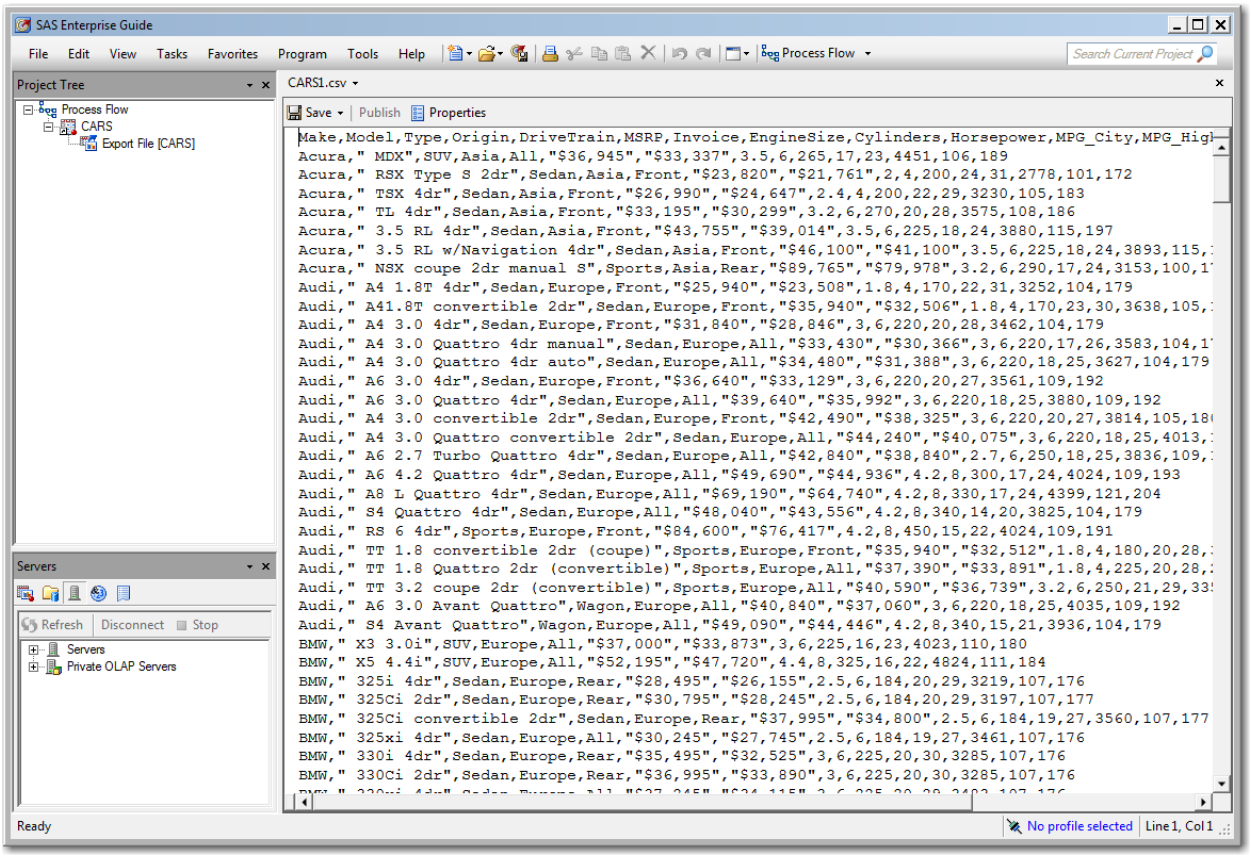
The **Results** tab displays the exported data.



Display 61 - Results Tab for Export File Node in SAS Enterprise Guide

Output Data Node for Export File in SAS Enterprise Guide

The Data node for the output data contains the exported data. You can edit and save the data in this node. Contrast this to the SAS Program Node for Output File in SAS Studio.



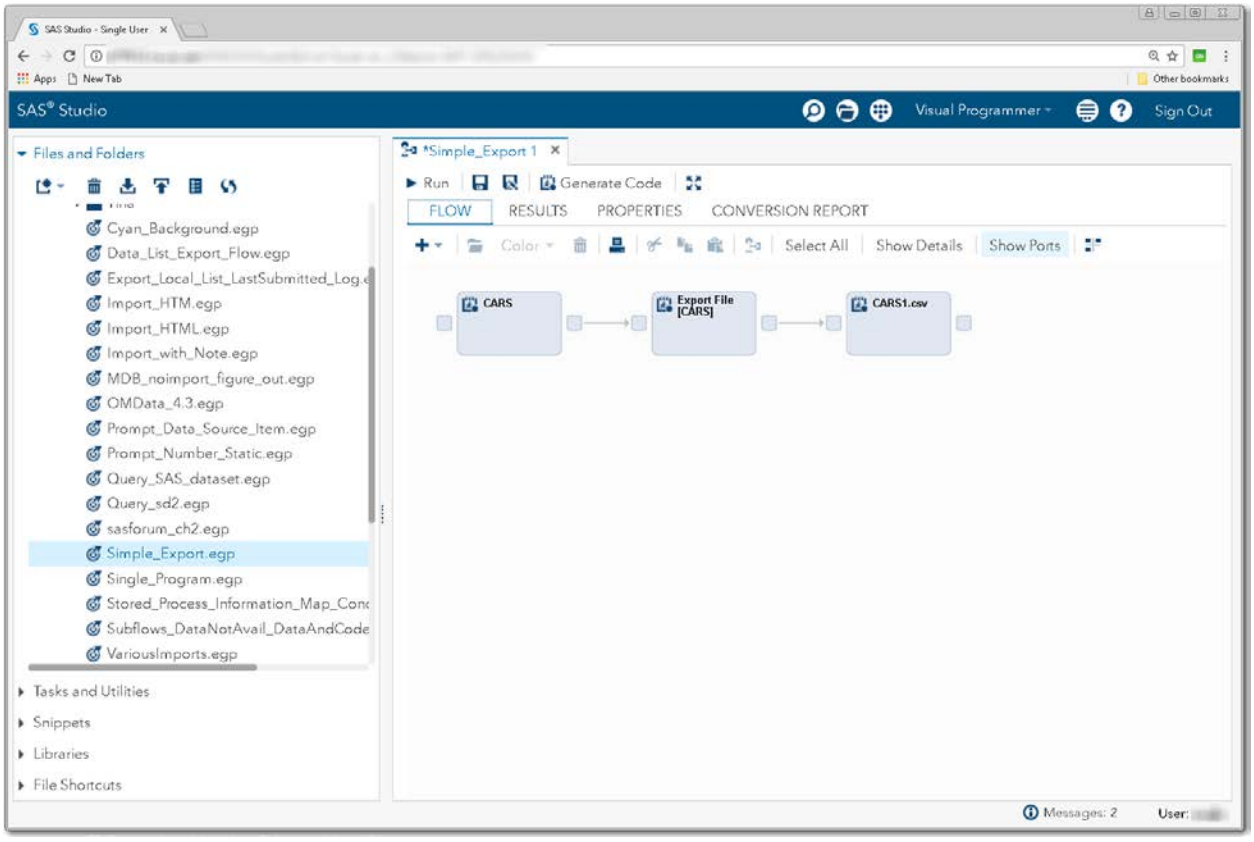
Display 62 - Contents of Node for Exported Data in SAS Enterprise Guide

SAS Studio

Converted Process Flow in SAS Studio

The process flow from SAS Enterprise looks very similar to the converted process flow in SAS Studio. While each of the nodes are represented in SAS Studio, these nodes are remarkably different in functionality.

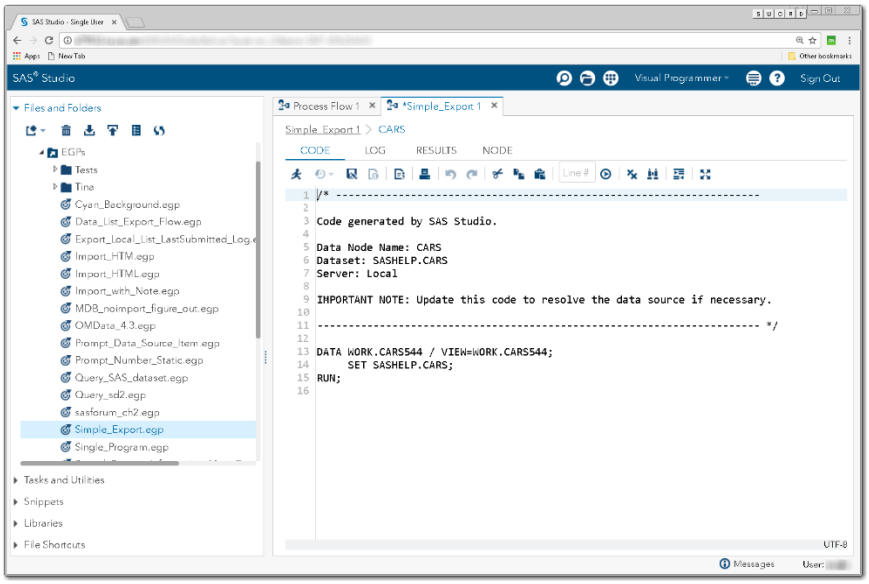
Contrast this display to the Process flow for Export File SAS in Enterprise Guide.



Display 63 - Process Flow for Converted Export File Node in SAS Studio

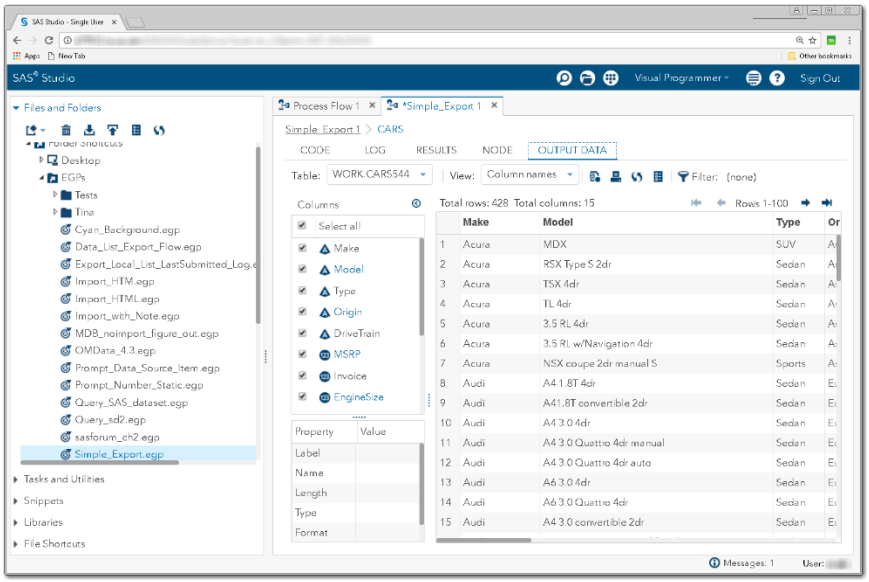
Data Set Node for Input Data in SAS Studio

The Data node for the input data set is converted to a SAS Program node that displays the contents of the SAS data set. You can contrast this to the Node for the Input SAS Data Set in SAS Enterprise Guide.



Display 64 - Replacement for Export Node of Input Data in SAS Studio

When you run this node in SAS Studio, the resulting **OUTPUT DATA** tab contains the contents of the SAS data set to export.

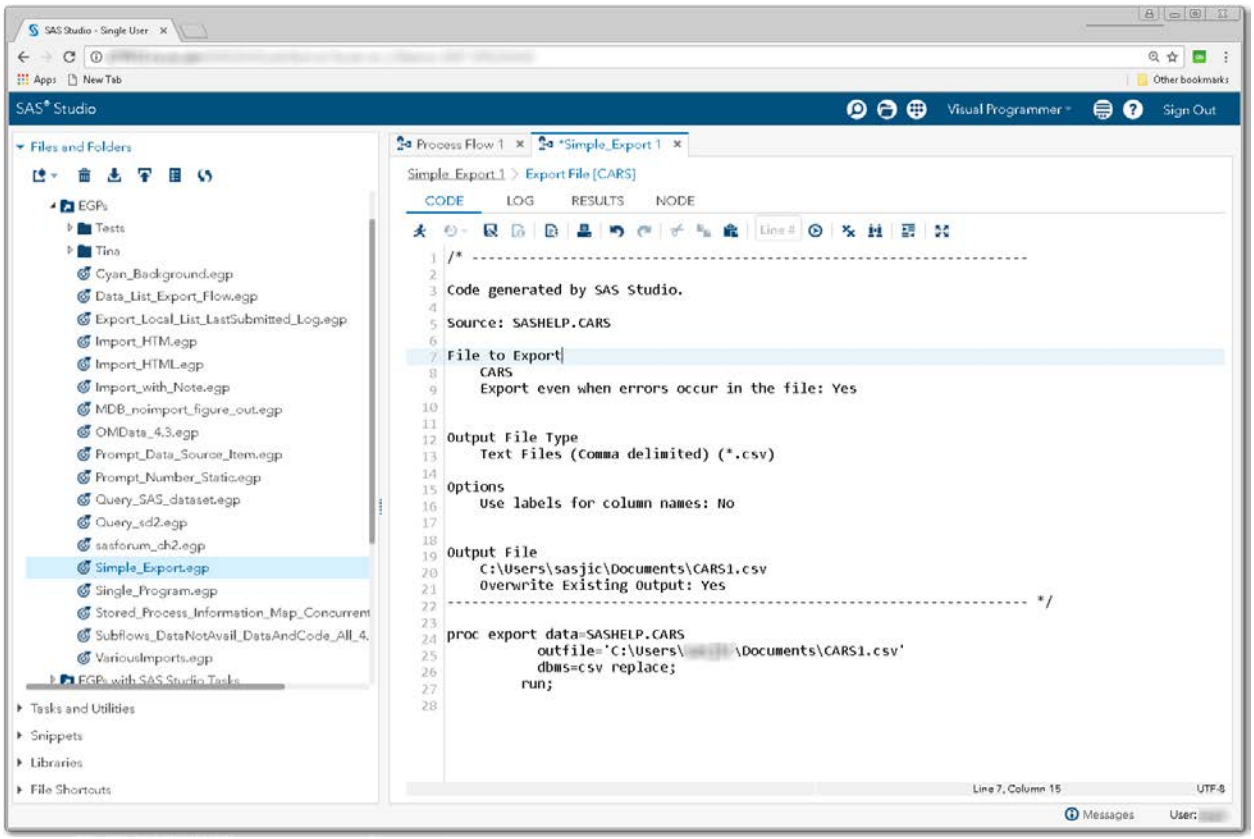


Display 65 - Results of Running the SAS Program Node Replacement for the Input Data Node in SAS Studio

Converted Export File Node in SAS Studio

Export File nodes in SAS Enteprise Guide are converted to SAS Program nodes in SAS Studio.

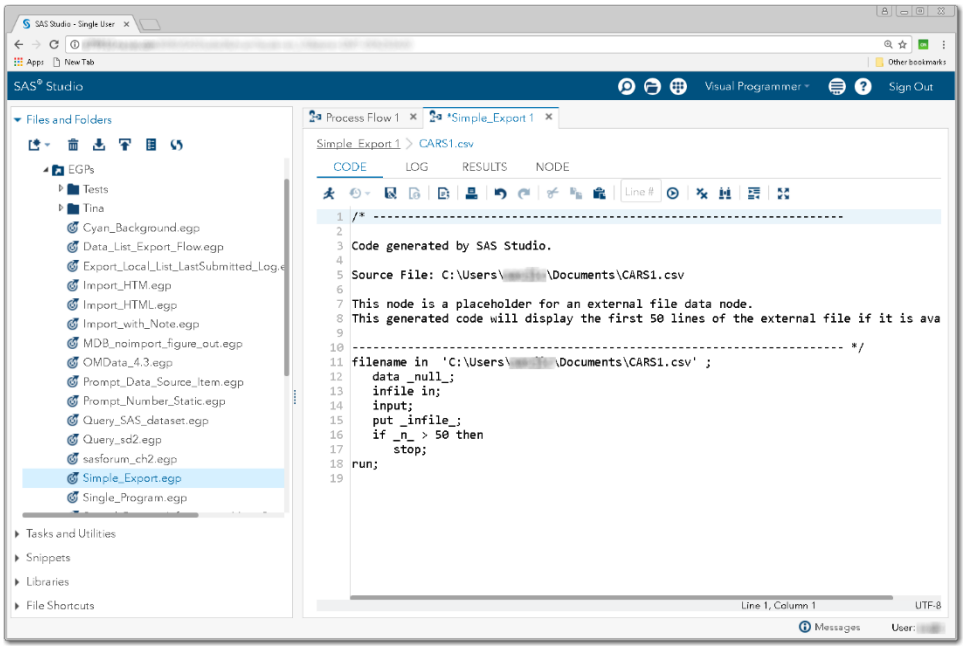
SAS Enterprise Guide does not generate portable export code in the Export File nodes. Therefore, SAS Studio uses the metadata that is stored in the SAS Enterprise Guide project to generate code that can export the input data set. If the output file location specified in the SAS Enterprise Guide process flow is not available using the active connection in SAS Studio, an error is written to the Conversion Report, but the code is still generated. Contrast this display to the Export File Node in SAS Enterprise Guide.



Display 66 - Converted Export File Node in SAS Studio

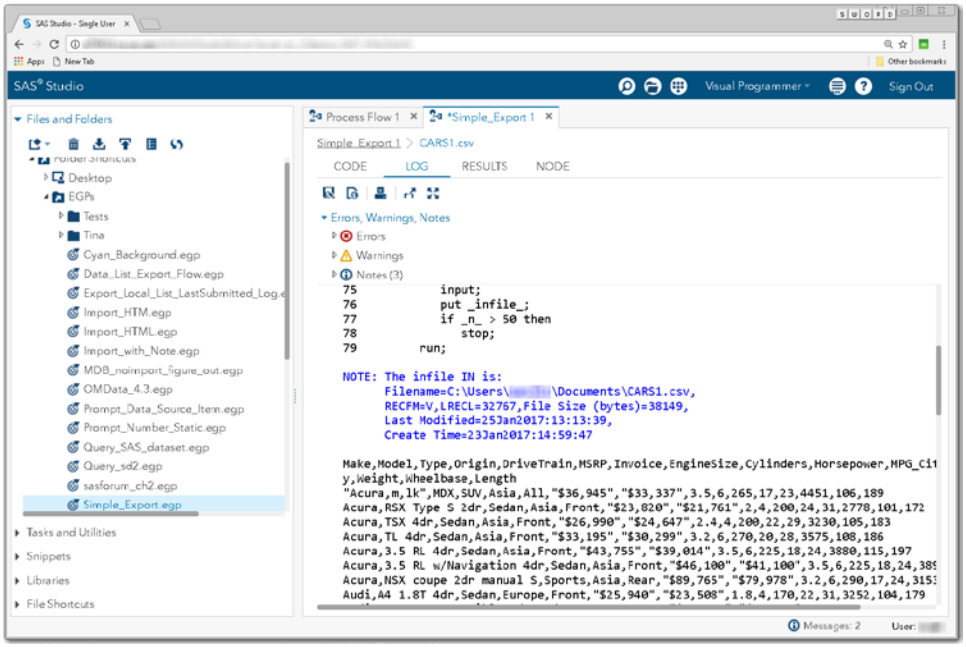
SAS Program Node for Output File in SAS Studio

The converted Data node for the output file becomes a data sampling SAS Program node. Contrast this to the Output Data Node for Export File in Enterprise Guide.



Display 67 - Generated Code for the Converted Data Node in SAS Studio

If you run the data sampling node, you will see up to 50 lines of your output file in the **LOG** tab.



Display 68 - Sampling the Output File

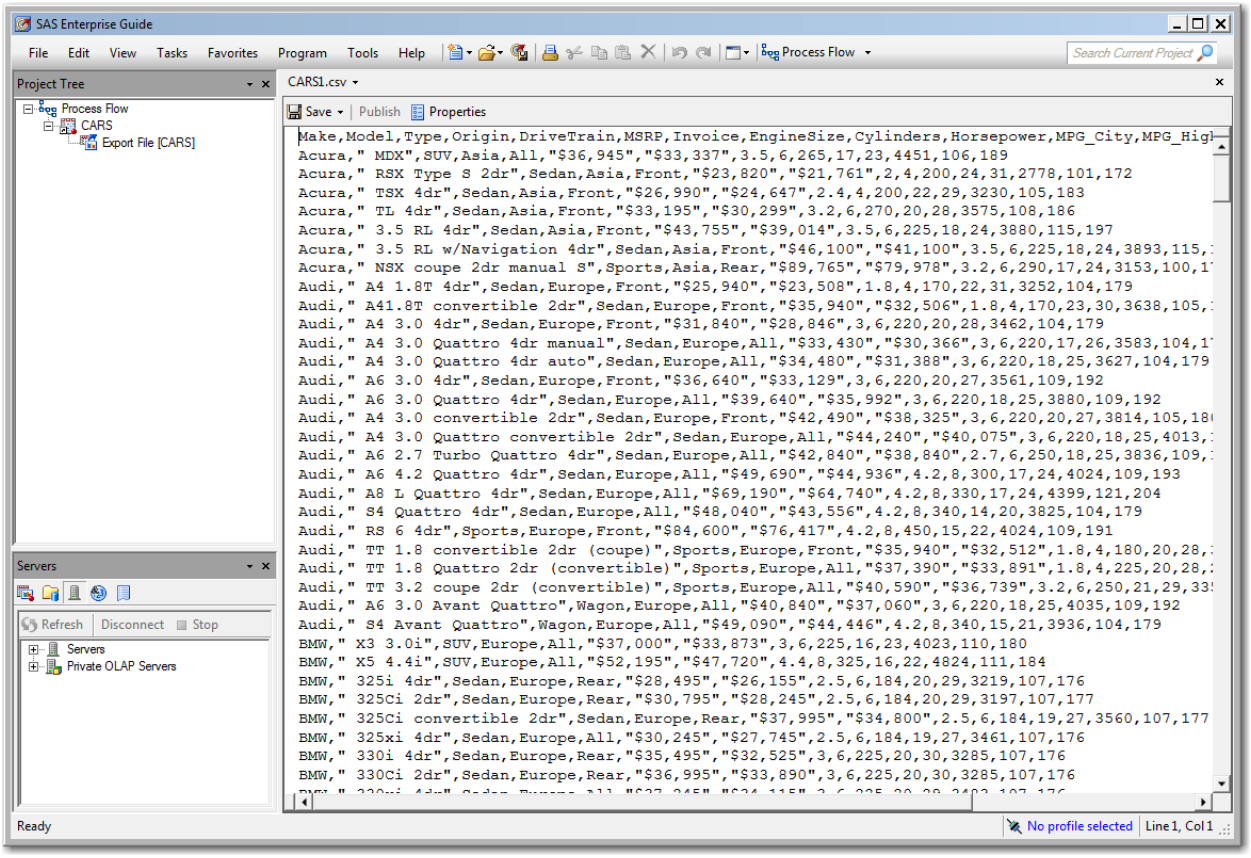
DATA NODES

Because process flows in SAS Studio 3.6 do not support Data nodes, the EGP conversion process converts them to SAS Program nodes.

The converted SAS Program node contains SAS code that:

- contains comments about the data location
- contains SAS code that samples the contents of the data in many cases. The generated code depends on the type of data in the Data node.

Contrast this functionality to Data nodes in SAS Enterprise Guide where you can open external files (excluding Microsoft Excel files) and edit them.



Display 69 Data Node for External File in SAS Enterprise Guide

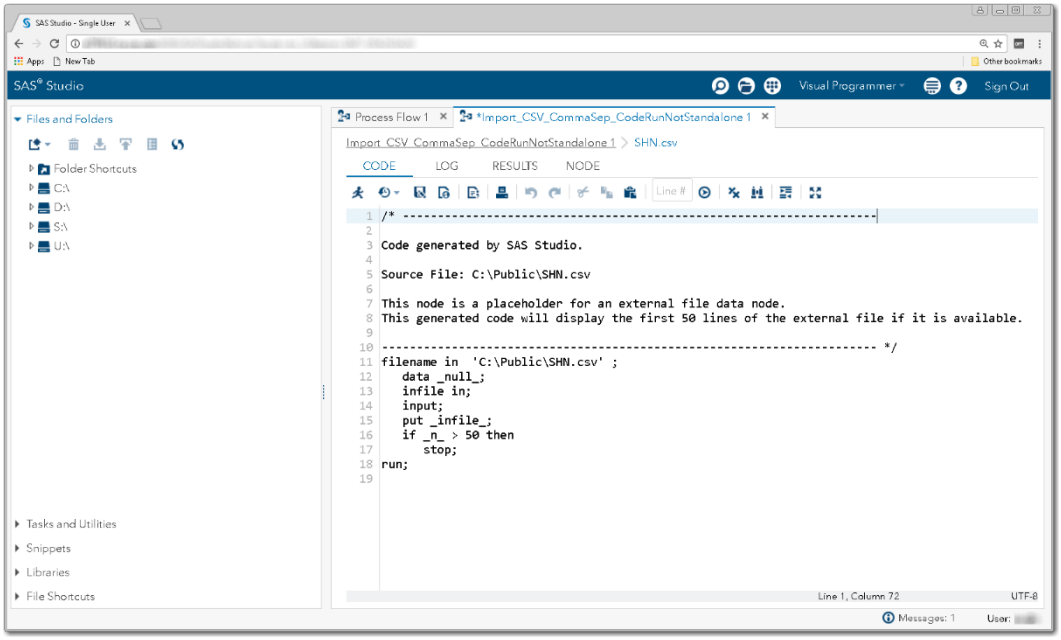
The current plan is for process flows in SAS Studio 5.1 to support Data nodes.

Data Sampling

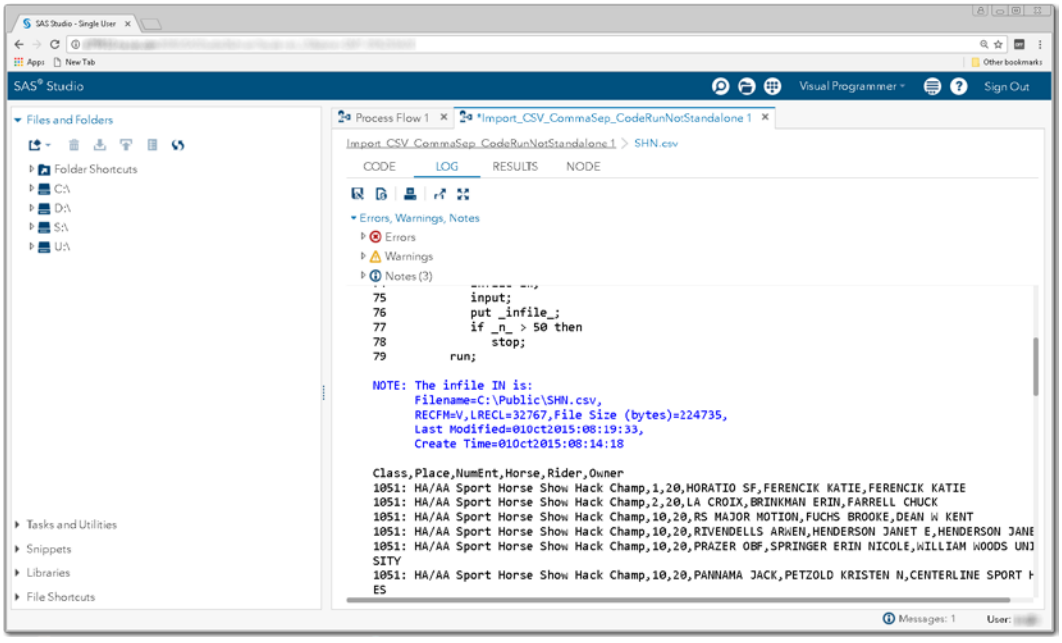
The first 50 lines of any directly readable text file, including CSV files, fixed length text files, HTML files, and tab delimited text files, are sampled. Data nodes that represent SAS data sets show a view of table rows.

Here are some examples of what you will see for these converted nodes.

CSV

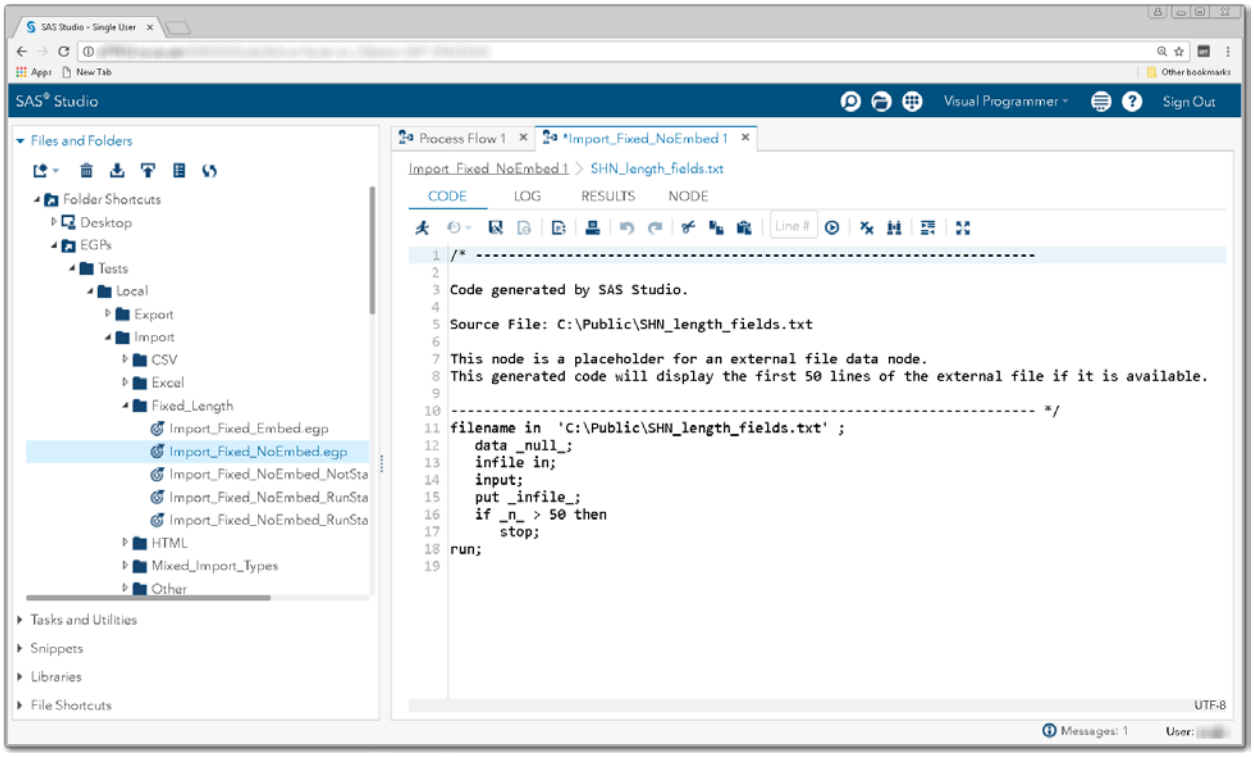


Display 70 - Converted Data Node for CSV File

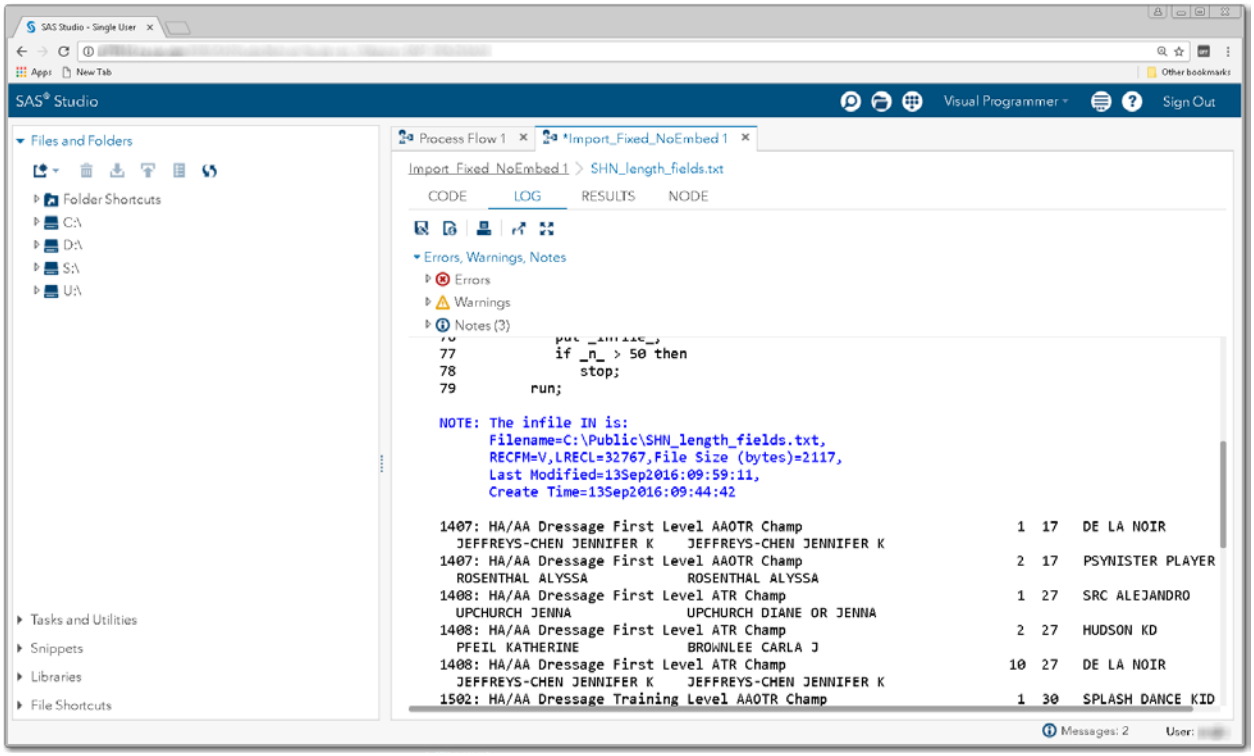


Display 71 - Log of Converted Data Node for CSV File

Fixed Length Text File

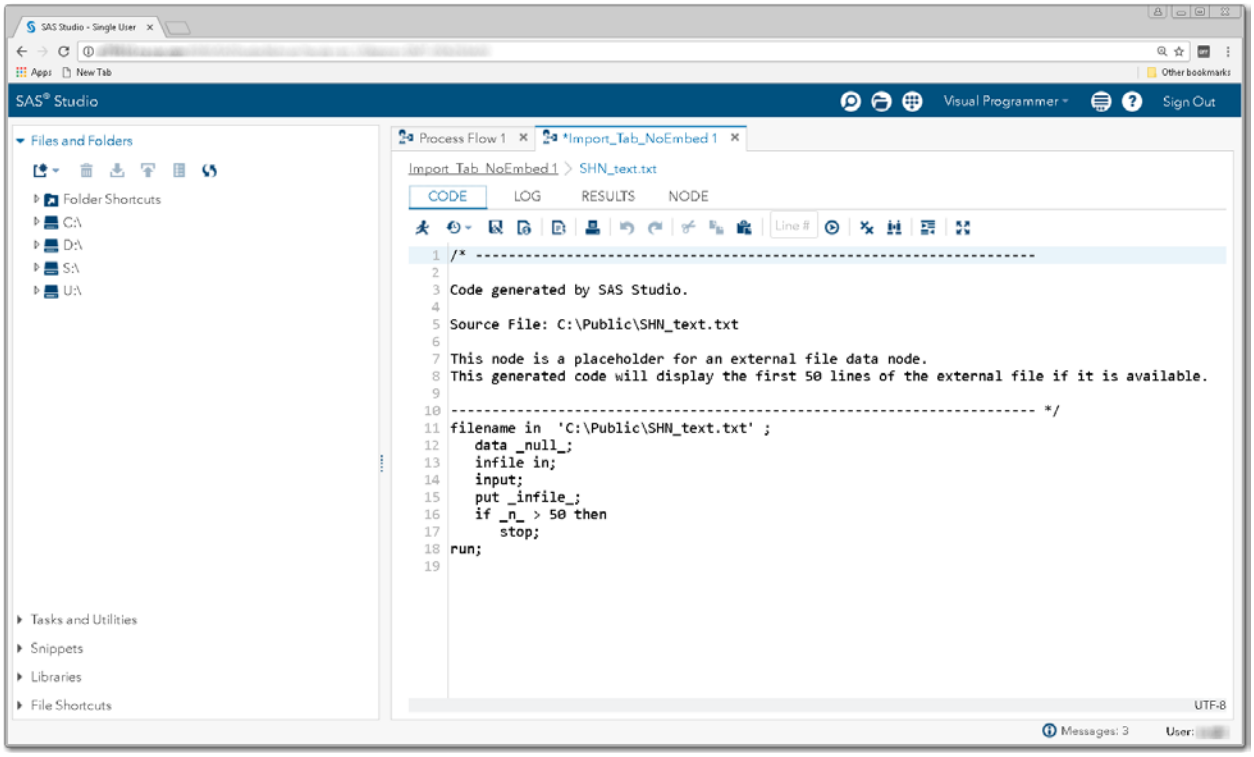


Display 72 - Converted Data Node for Fixed Length Text File

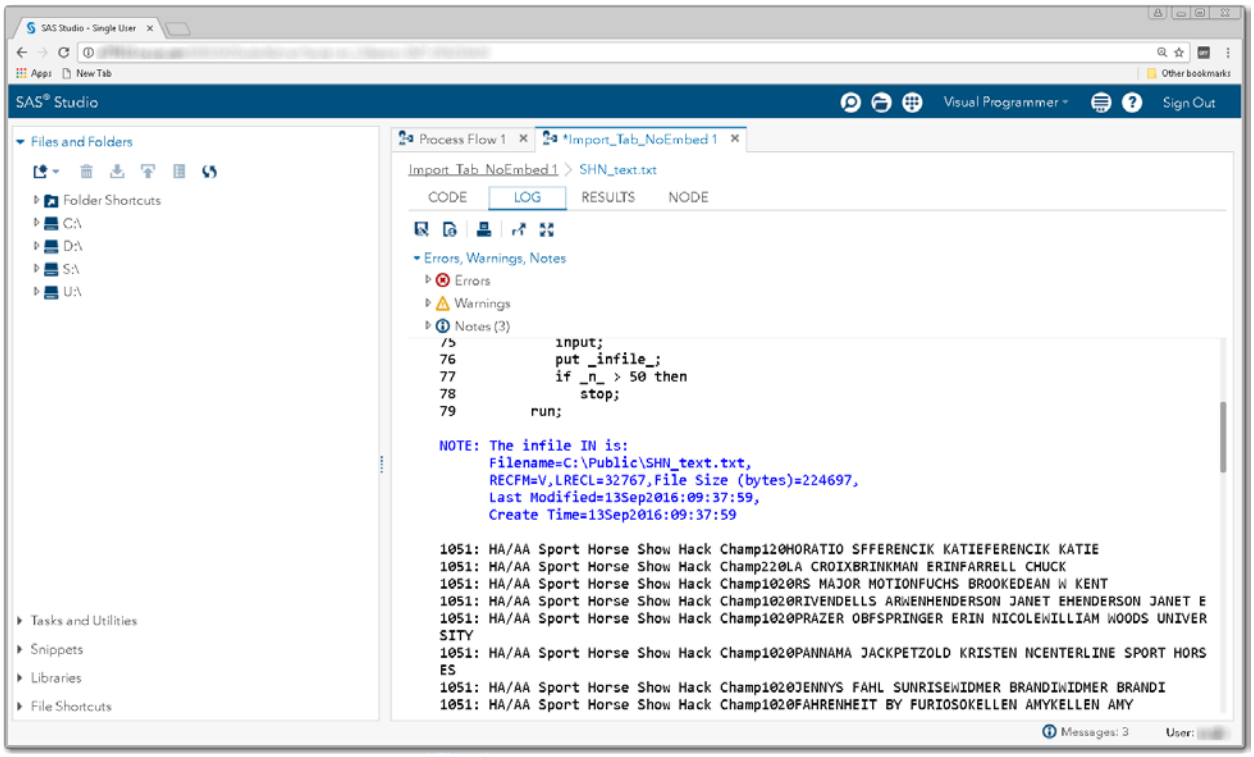


Display 73 - Log for Converted Data Node for Fixed Length Text File

Tab Delimited Text File



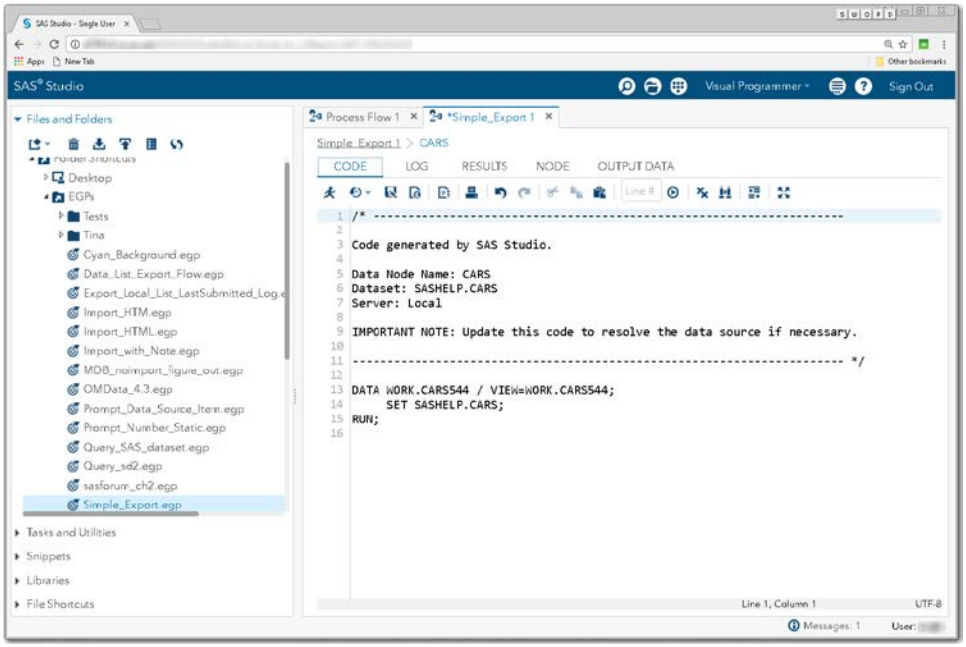
Display 74 - Code for Converted Data Node for Tabbed Text File



Display 75 - Log for Converted Data Node for Tabbed Text FileSAS Data Set Nodes in

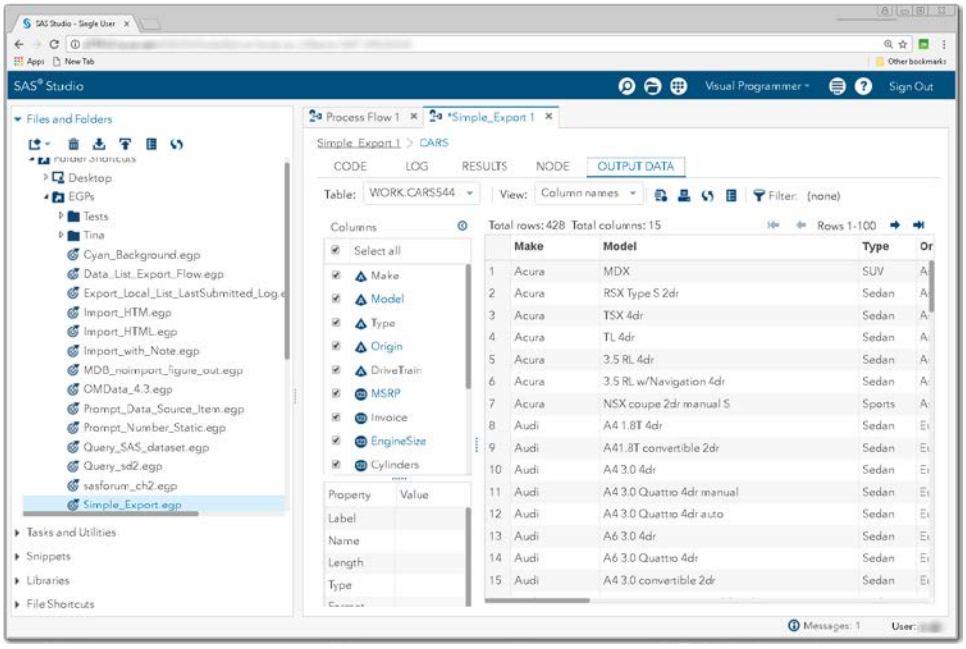
SAS Studio

SAS Studio converts Data nodes for SAS data sets to SAS Program nodes. These nodes contain code that displays a view of the table.



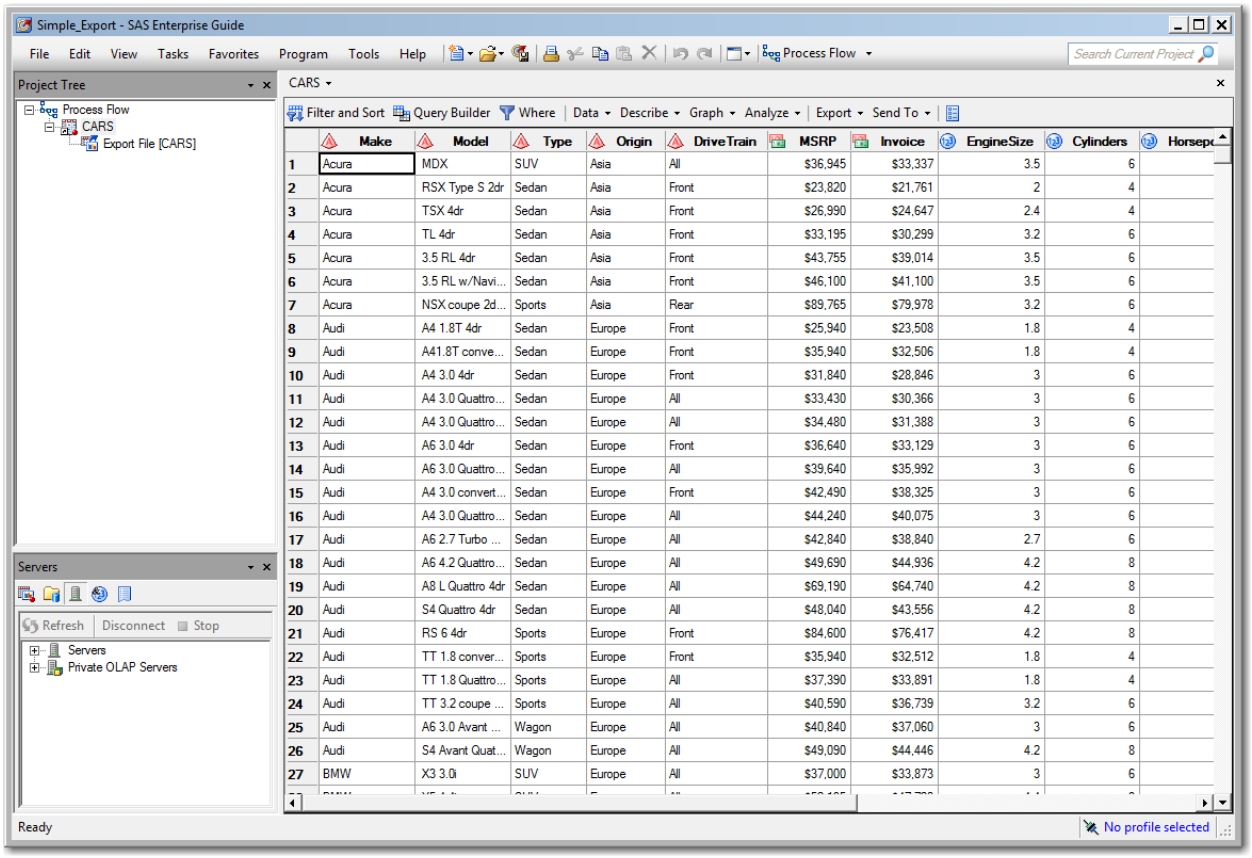
Display 76 - Code for Converted Data Node for SAS Data Set

When you run the process flow or the converted Data node for the SAS data set in the SAS Studio process flow, the **OUTPUT DATA** tab displays a view of the table. You cannot edit the table or perform any other functions in this view.



Display 77 - Output from Converted Data Node in SAS Studio

In SAS Enterprise Guide, you can also see the table contents of this node. However in SAS Enterprise Guide, you can perform many other functions including editing the contents of the table.



	Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize	Cylinders	Horsepc
1	Acura	MDX	SUV	Asia	All	$36,945	$33,337	3.5	6	
2	Acura	RSX Type S 2dr	Sedan	Asia	Front	$23,820	$21,761	2	4	
3	Acura	TSX 4dr	Sedan	Asia	Front	$26,990	$24,647	2.4	4	
4	Acura	TL 4dr	Sedan	Asia	Front	$33,195	$30,299	3.2	6	
5	Acura	3.5 RL 4dr	Sedan	Asia	Front	$43,755	$39,014	3.5	6	
6	Acura	3.5 RL w/Navi...	Sedan	Asia	Front	$46,100	$41,100	3.5	6	
7	Acura	NSX coupe 2d...	Sports	Asia	Rear	$89,765	$79,978	3.2	6	
8	Audi	A4 1.8T 4dr	Sedan	Europe	Front	$25,940	$23,508	1.8	4	
9	Audi	A41.8T conve...	Sedan	Europe	Front	$35,940	$32,506	1.8	4	
10	Audi	A4 3.0 4dr	Sedan	Europe	Front	$31,840	$28,846	3	6	
11	Audi	A4 3.0 Quattro...	Sedan	Europe	All	$33,430	$30,366	3	6	
12	Audi	A4 3.0 Quattro...	Sedan	Europe	All	$34,480	$31,388	3	6	
13	Audi	A6 3.0 4dr	Sedan	Europe	Front	$36,640	$33,129	3	6	
14	Audi	A6 3.0 Quattro...	Sedan	Europe	All	$39,640	$35,992	3	6	
15	Audi	A4 3.0 convert...	Sedan	Europe	Front	$42,490	$38,325	3	6	
16	Audi	A4 3.0 Quattro...	Sedan	Europe	All	$44,240	$40,075	3	6	
17	Audi	A6 2.7 Turbo ...	Sedan	Europe	All	$42,840	$38,840	2.7	6	
18	Audi	A6 4.2 Quattro...	Sedan	Europe	All	$49,690	$44,936	4.2	8	
19	Audi	A8 L Quattro 4dr	Sedan	Europe	All	$69,190	$64,740	4.2	8	
20	Audi	S4 Quattro 4dr	Sedan	Europe	All	$48,040	$43,556	4.2	8	
21	Audi	RS 6 4dr	Sports	Europe	Front	$84,600	$76,417	4.2	8	
22	Audi	TT 1.8 conver...	Sports	Europe	Front	$35,940	$32,512	1.8	4	
23	Audi	TT 1.8 Quattro...	Sports	Europe	All	$37,390	$33,891	1.8	4	
24	Audi	TT 3.2 coupe ...	Sports	Europe	All	$40,590	$36,739	3.2	6	
25	Audi	A6 3.0 Avant ...	Wagon	Europe	All	$40,840	$37,060	3	6	
26	Audi	S4 Avant Quat...	Wagon	Europe	All	$49,090	$44,446	4.2	8	
27	BMW	X3 3.0i	SUV	Europe	All	$37,000	$33,873	3	6	

Display 78 – Data Node for SAS Data Set in SAS Enterprise Guide

Data Nodes Not Sampled in SAS Studio

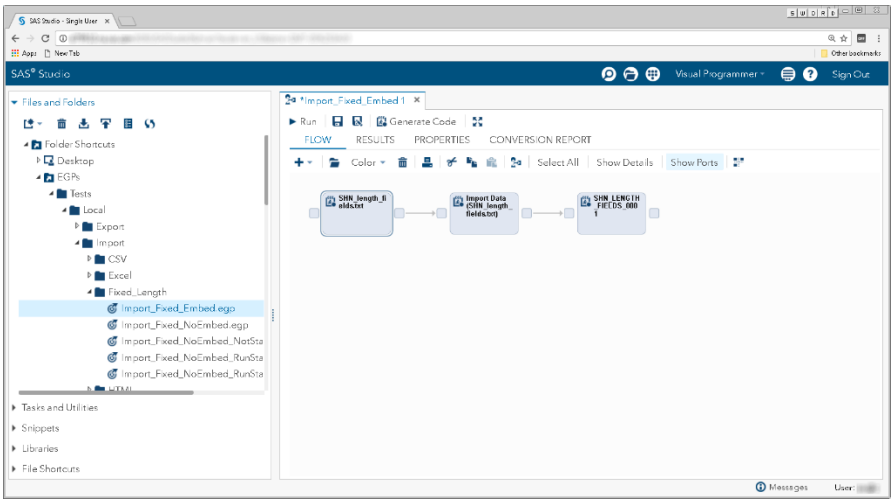
The contents of the Data node are not sampled in the converted node when:

- the EGP was configured to have data embedded in the generated SAS code. In this case, the data in the file is not used by the process flow. It is embedded in the task that uses the data.
- the data source is an Excel file. Excel data would have to be extracted to sample.

Data Is Embedded in the SAS Code

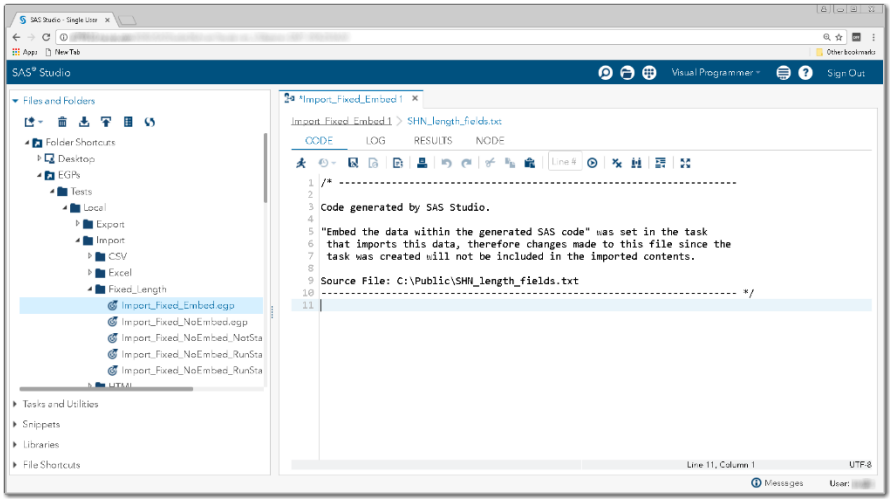
If you selected the **Embed the data within the generated SAS code** option, the converted node will contain just a comment.

The following converted process flow contains a Program File node that is used to identify data that is embedded into the Import Data node. As a result, the process flow in SAS Studio is visually similar to the one in SAS Enterprise Guide.



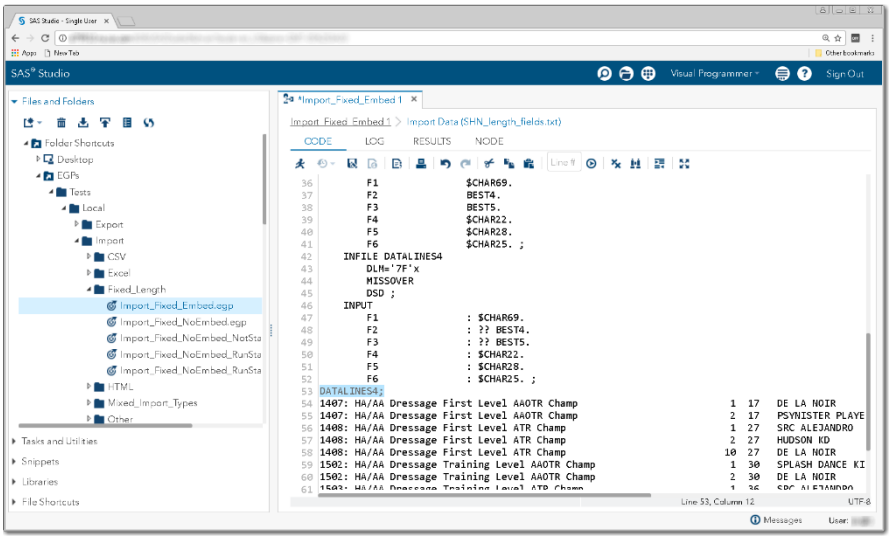
Display 79 - Contents of Source Data Are Embedded in Import Node

When you open the converted Data node, you see comments that provide the source location of the data that is embedded in the Import Data node. These comments also explain that updating the source data will not change the outcome of running the program.



Display 80 - Comments about the Embedded Data

If you open the Import Data node, you see the data from the input file in the DATALINES statement.

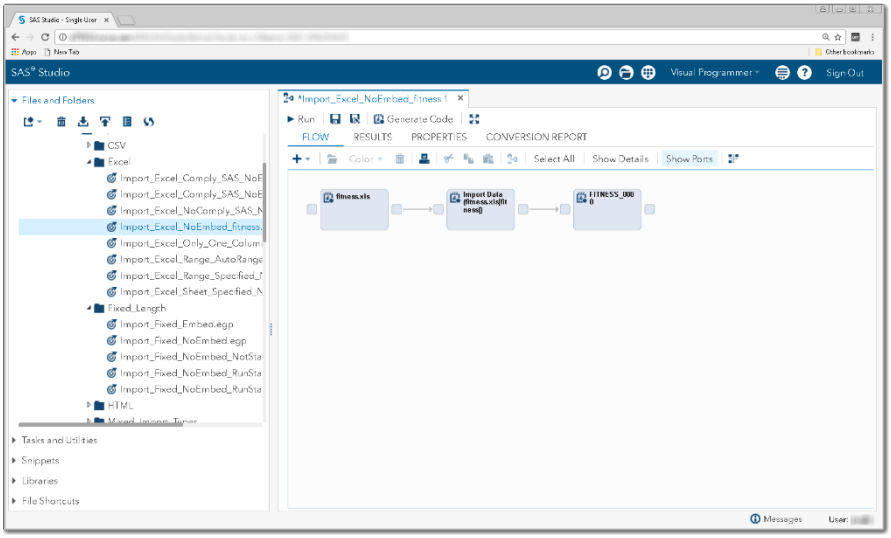


Display 81 - Data Embedded in Converted Import Node

Data Is from Excel

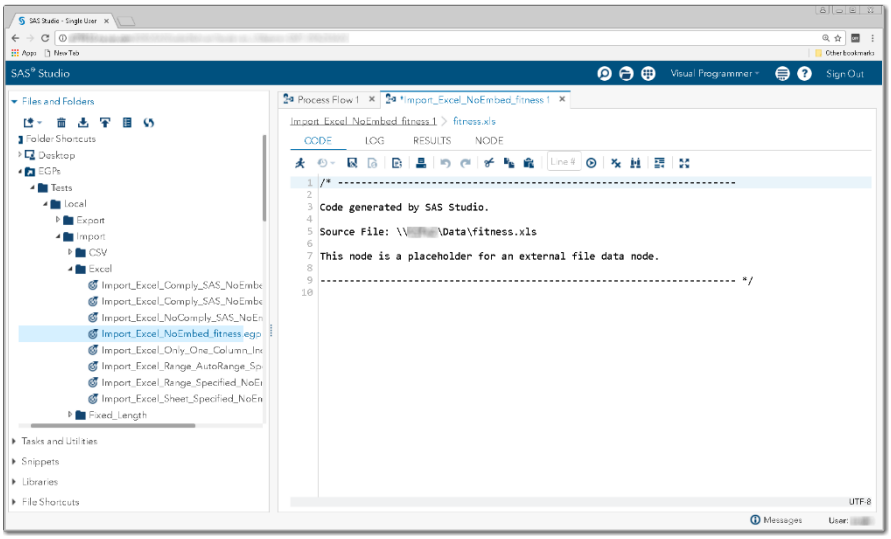
If the source for the Data node is an Excel file, the Data node is converted to a SAS Program node. The code in the converted SAS Program node in SAS Studio will not sample the contents of the Excel file because this sampling would require importing the file contents. The converted node contains only comments identifying the location of the Excel file. This node exists in the converted flow only to make the process flow look similar to how it looks in SAS Enterprise Guide.

In this flow, an Excel file is the data source for the Import Data node.



Display 82 - Data Node for Excel Data

The contents of the converted Data node for an Excel File is simply comments.



Display 83 - Converted Data Node When Input Data Is an Excel File

SAS ENTERPRISE GUIDE TASKS

Since SAS Enterprise Guide tasks do not port directly to SAS Studio, code generated by a SAS Enterprise Guide Task node is extracted and placed in a SAS Program node in SAS Studio.

The SAS Enterprise Guide Task nodes are associated with a particular SAS connection. If the SAS Studio connection does not match the connection associated with the Task node in SAS Enterprise Guide, an error appears in the log.

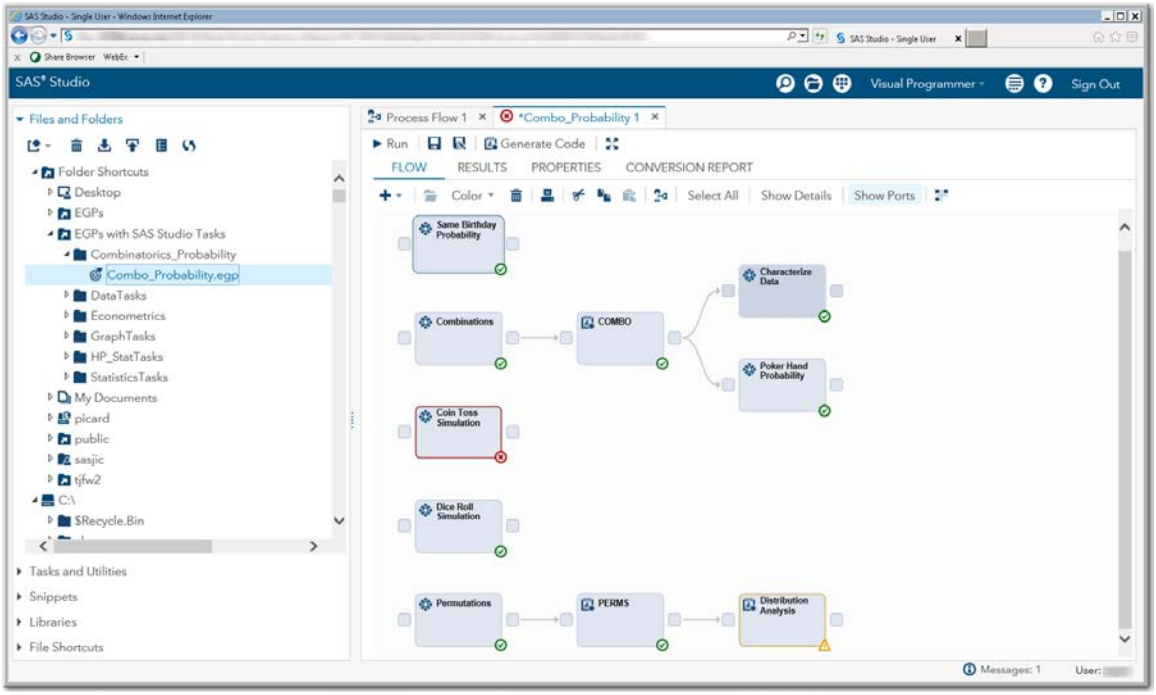
QUERY BUILDER NODES

Since SAS Enterprise Guide Query Builder nodes do not port directly to SAS Studio, code generated by a SAS Enterprise Guide Query Builder node is extracted and placed in a SAS Program node in SAS Studio.

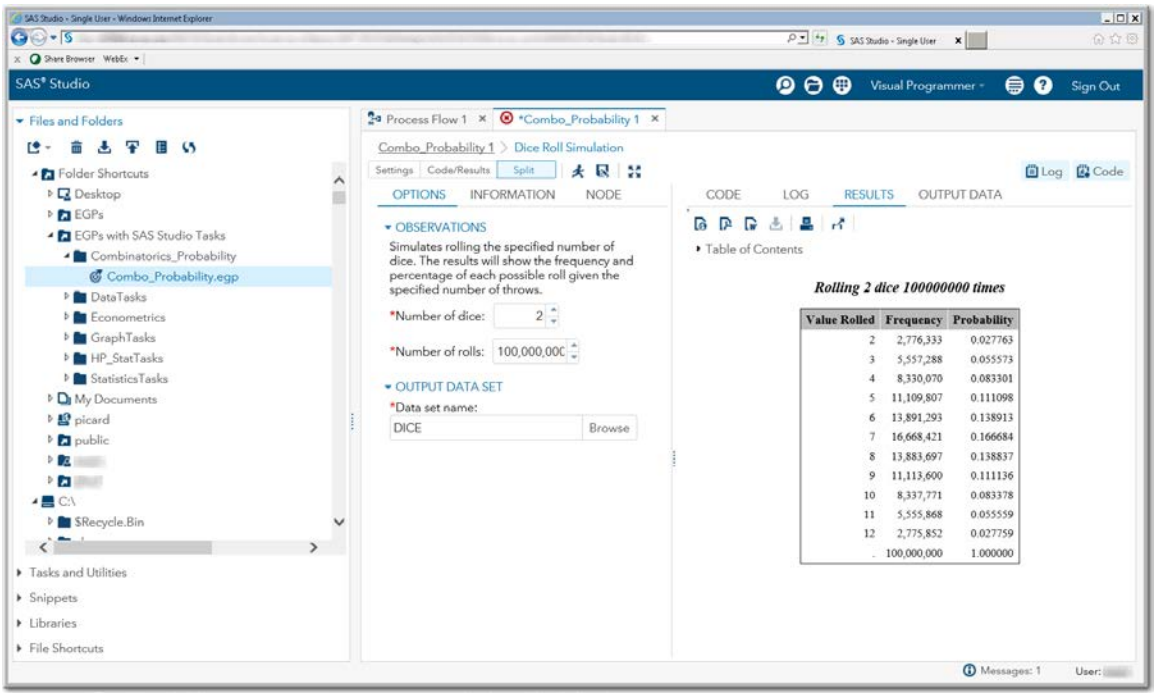
SAS STUDIO TASKS

SAS Studio tasks are also available in SAS Enterprise Guide. SAS Studio Task nodes convert seamlessly. Any parameters set on a SAS Studio Task node will be in the converted node.

Again, server context is very important since the code for the SAS Studio tasks are version dependent.



Display 84 - Converted Process Flow That Contains SAS Studio Tasks



Display 85 – User Interface and Results for SAS Studio Task Node

PROMPTS

SAS Enterprise Guide

In SAS Enterprise Guide when a process flow containing a program with prompts or a prompt dependent program is run, a dialog box that contains prompts is displayed.

When the user enters values in the prompt dialog box, these values are placed in macro variables with names based on the **Name** field in the prompt definition. Some prompts, such as a simple text or numeric input field, create a single macro variable with the prompt name. Other prompts such as a text range or a numeric range create multiple macro variables, such as Name_min and Name_max. Multiple nodes can depend on a prompt.

SAS Studio

SAS Studio does not support prompts. When an EGP that contains prompts is converted, code is added to the beginning of the converted Program nodes that have prompts associated with them. The added code will:

- contain comments about the values expected for the macro variables, such as types, minimums and maximums, and so on.
- define macro variables that the prompts would have created.
- set the macro variables to:
    - default values if the prompts have defaults
    - blank or empty if prompts do not have defaults
- delete the macro variables at the end of the program unless the prompt definition has the **Use prompt value throughout project** option selected. (You select this check box in the Prompt Manager in SAS Enterprise Guide.)

If you want to run your prompt dependent code against different parameters, you must manually change the values in the generated macro variables.

If a prompt is used for more than one node, the values of the macro variables would have to be manually changed in each node to simulate the prompt. This editing can be very tedious if there are many nodes depending on a single prompt.

If you would prefer to change the prompt values with a user interface, you can use a SAS Studio task as a prompt replacement.

SAS Studio Task as Prompt Replacement

You can manually update a converted process flow to use a SAS Studio task to collect the information that the prompts would have collected. To do this, you need to:

- write a SAS Studio task that contains input controls for each of the macro variables that the prompts would have created. The prompt replacement code generated by SAS Studio provides information about the macro variables and the attributes of the macro variables that are defined by the prompts.
- add the SAS Studio task to your converted process flow.
- create links from the output port of the SAS Studio task to the input ports of the SAS Program nodes that depend on the prompt or prompts.
- in the converted Program nodes that depend on the prompts, comment out the call to the macro code that creates and sets global variables. The %SYMDEL code in single input cases can remain.

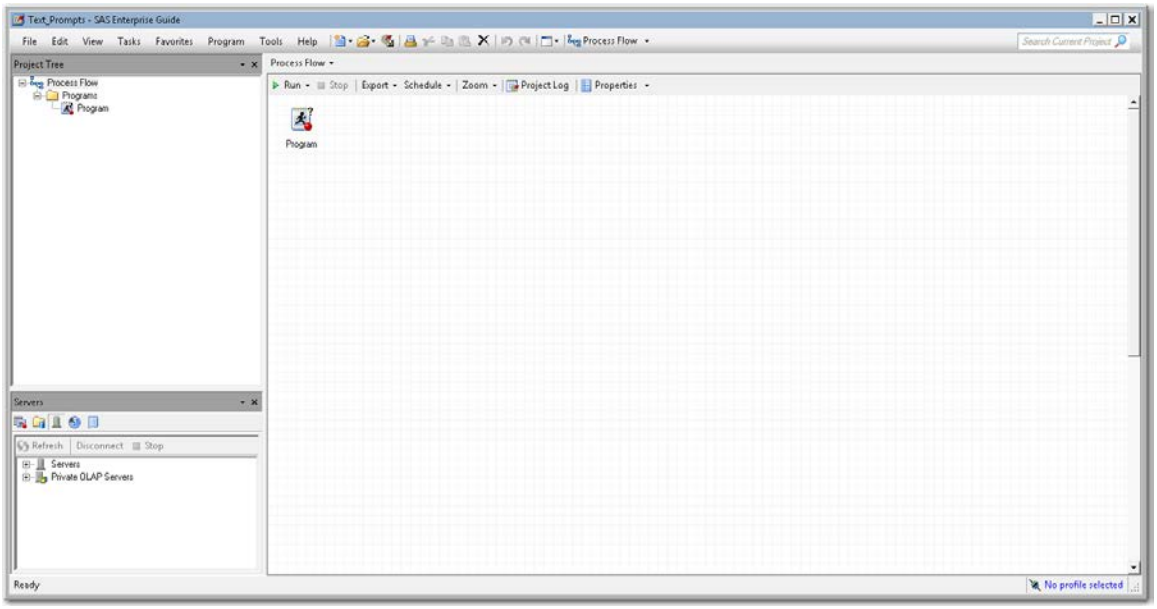
When using a SAS Studio task to simulate a SAS Enterprise Guide prompt, the task functions like a SAS Enterprise Guide prompt that has **Hide at Runtime** option selected. If you would like to run with different values, you must open the SAS Studio task, change the values, and then run the process flow or program.

For more information about how to write a task, see *SAS Studio 3.6: Developer's Guide to Writing Custom Tasks* . A good starting point for writing a task is to view the Sample Task in SAS Studio. From the **Tasks and Utilities** section, click the New icon and select **Sample Task**. The Sample Task opens in a Task Editor window. Press the **Run** button to see the controls that the task creates. Then you can use the **New Task** option to create a new task and add the controls that you need for requesting the prompt input. Use the Sample Task as a resource.

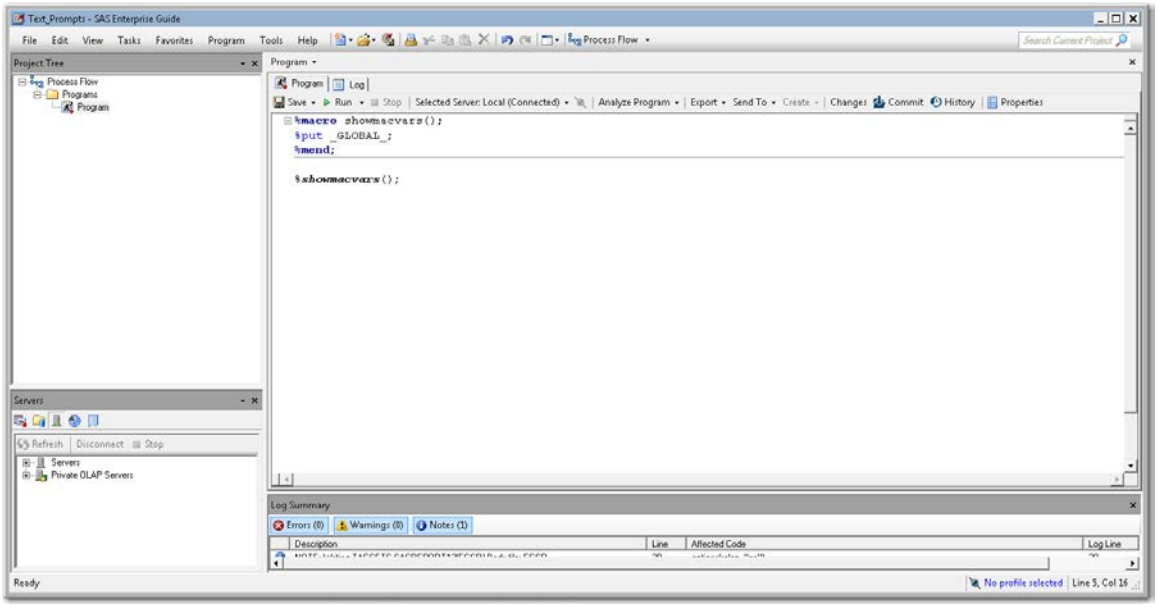
The following sections provide examples of the types of prompts supported by SAS Enterprise Guide and describe the macro variables that are generated for those types. Then the same examples are shown converted in SAS Studio with information about the generated macro code for prompt replacement in the SAS Program node. For many of the prompt types, there are explanations on how to use SAS Studio tasks to replace the prompts. Not all of the range and multiple selection prompt types have an example, but you can get the idea about how to code ranges for those types from the other range and multiple selection examples. All of the examples can be further enhanced to be more complete. For example, the range tasks could have error checks for minimums that are greater than maximums.

Program Using Sample Prompt Definitions

The following program is used for all of the prompt examples. This program displays the contents of the global variables.



Display 86 - Program Node in SAS Enterprise Guide



Display 87 – Code in the Program Node in SAS Enterprise Guide

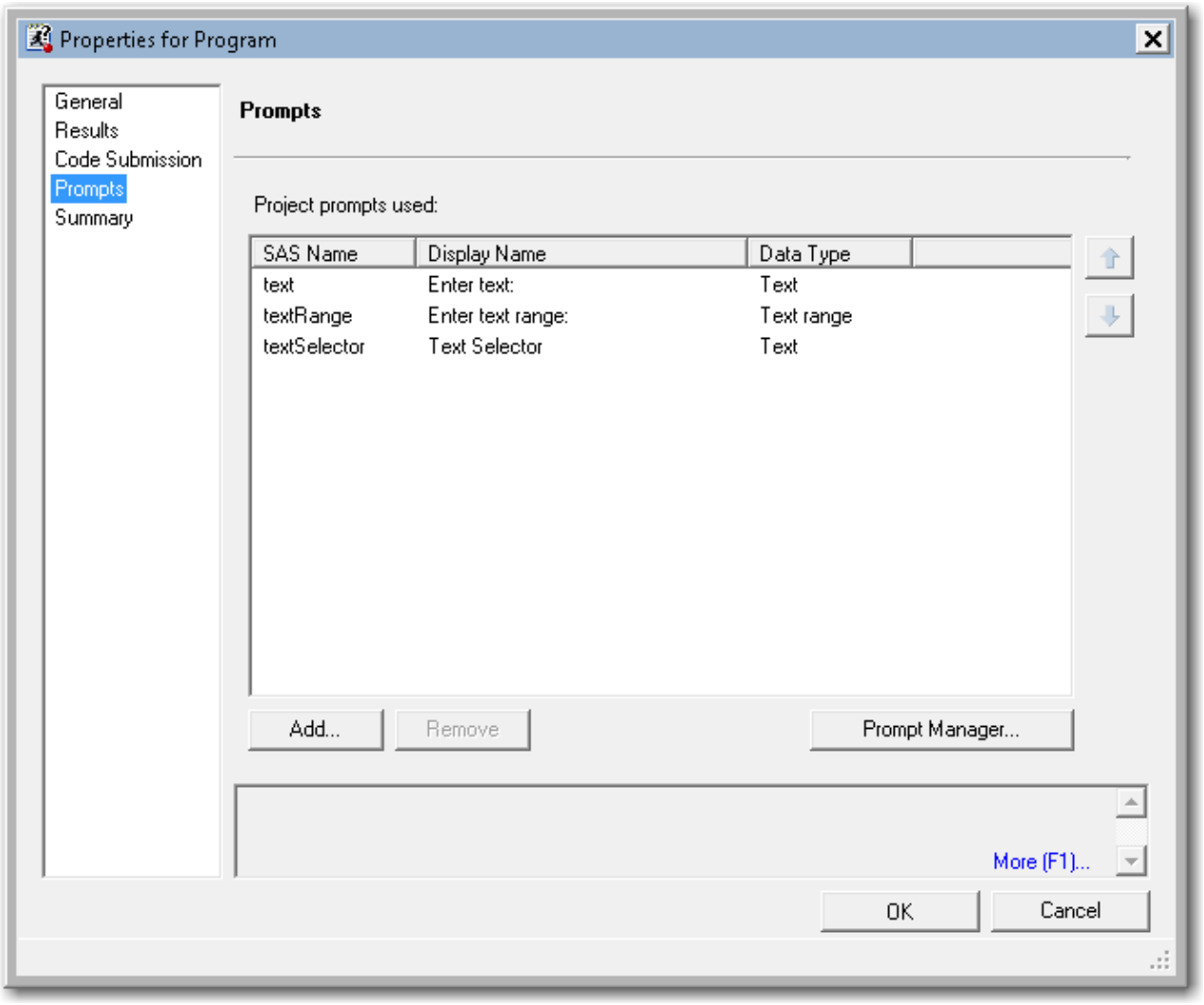
In SAS Enterprise Guide to define a prompt, right-click the Program node and select **Properties**. From the selection pane, select **Prompt** and click **Prompt Manager**. You define prompts in the Prompt Manager.

After you have defined your prompts, you can select the prompts to add to your Program nodes by clicking the **Add** button.

Text

The name of the macro variable or variables defined for a text prompt depends on whether it is a single value, a text range, or multiple text value prompt.

The following example contains the three different types of text prompts and describes the macro variables that are defined by SAS Enterprise Guide for each prompt type.

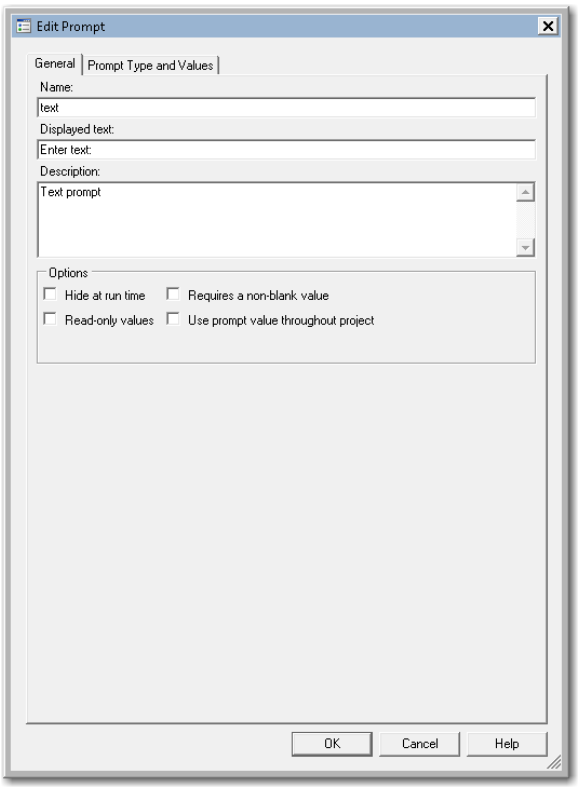


Display 88 - Text Prompts Defined in SAS Enterprise Guide

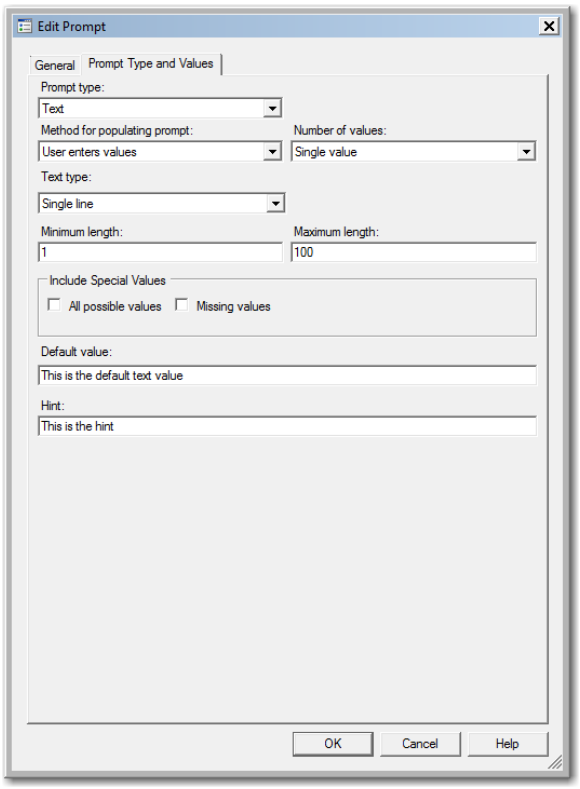
Single Value Text

SAS Enterprise Guide

In this example, a single value text prompt named **text** is defined.

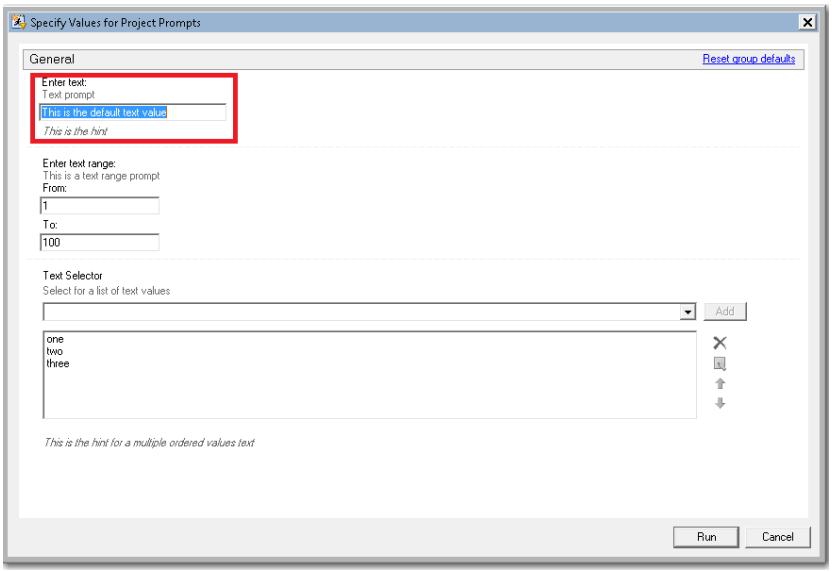


Display 89 - General Attributes for Text Prompt



Display 90 - Prompt Type and Values for Text Prompt

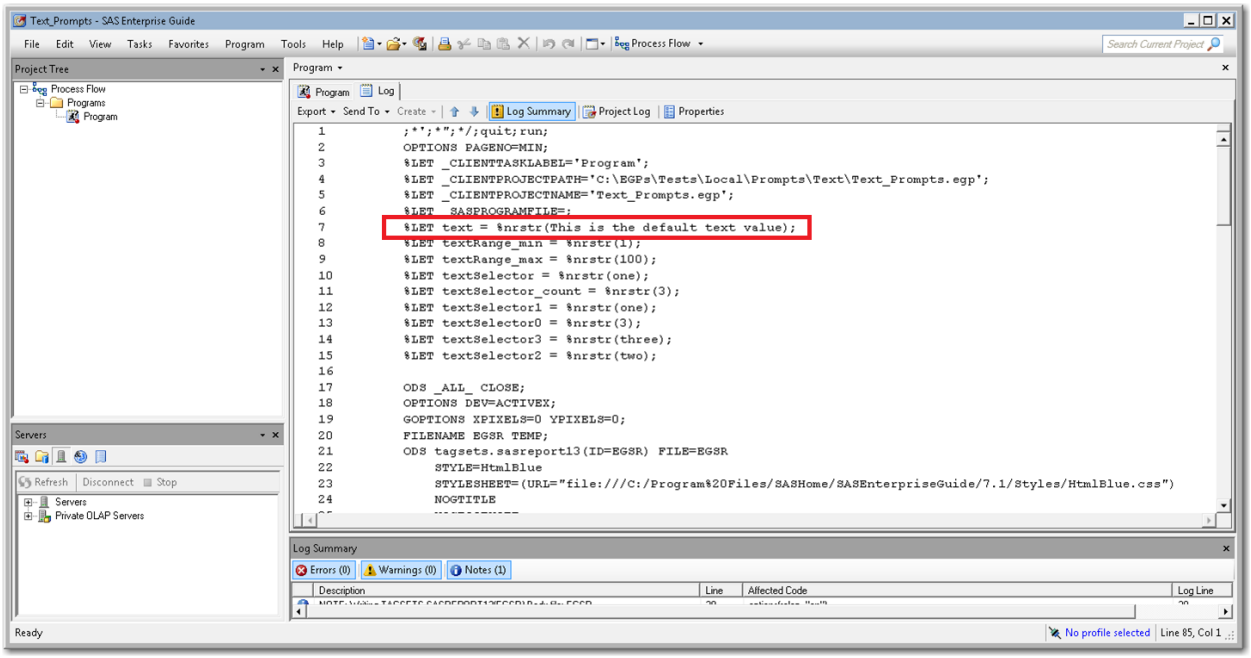
When you run the Program node that depends on this prompt, the following dialog box appears.



Display 91 - Single Text Value Prompt in Prompt Dialog Box

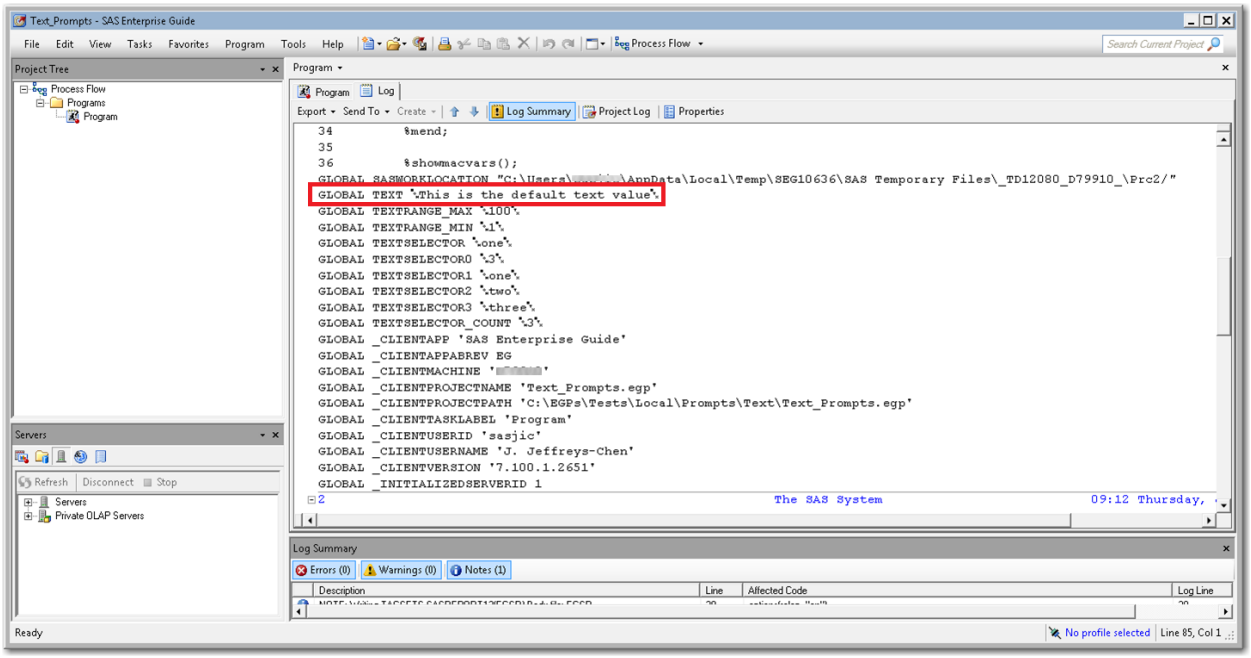
If the user leaves the default value in the single text value prompt field, the following code is generated by SAS Enterprise Guide in the Program node that depends on the prompt.

A %LET statement assigns the value specified in the prompt dialog box to the text macro variable.



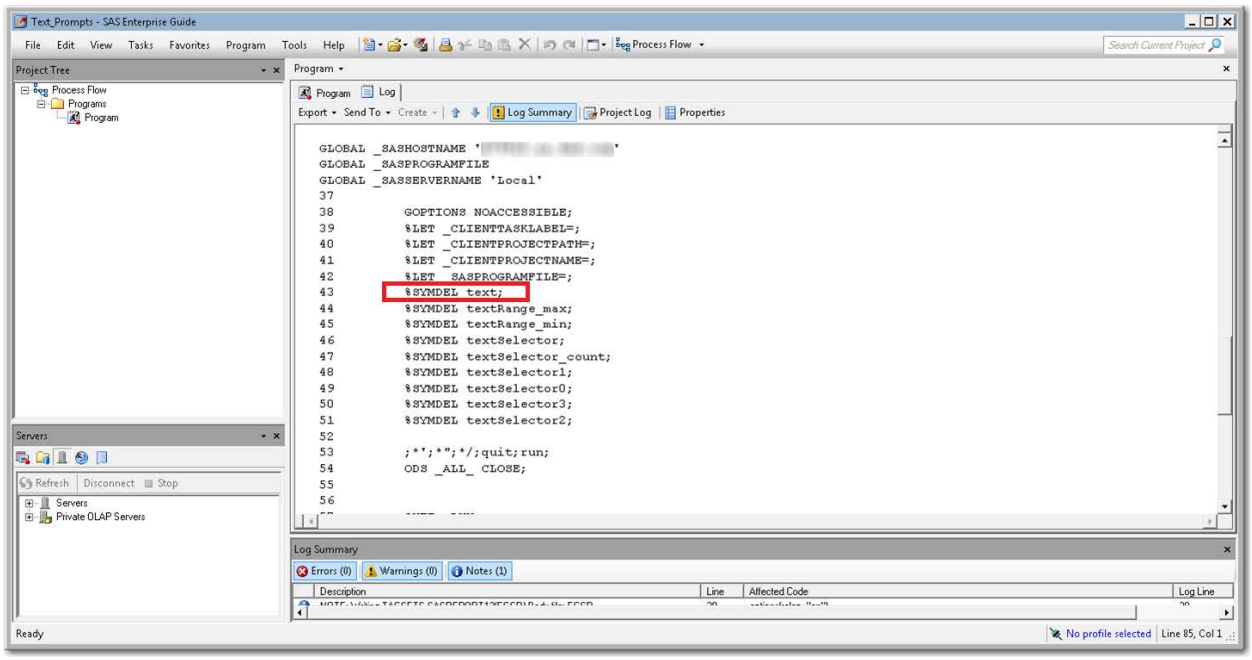
Display 92 - Assignment of %LET Statements for Single Value Text Prompt

The log of the Program node using the prompt definition displays the value of the global variable created by the prompt.



Display 93 - Global Definition for Text Macro Variable

Because the **Use prompt value throughout project** option is not checked for this prompt, the %SYMDEL statement removes the text macro variable at the end of the program.

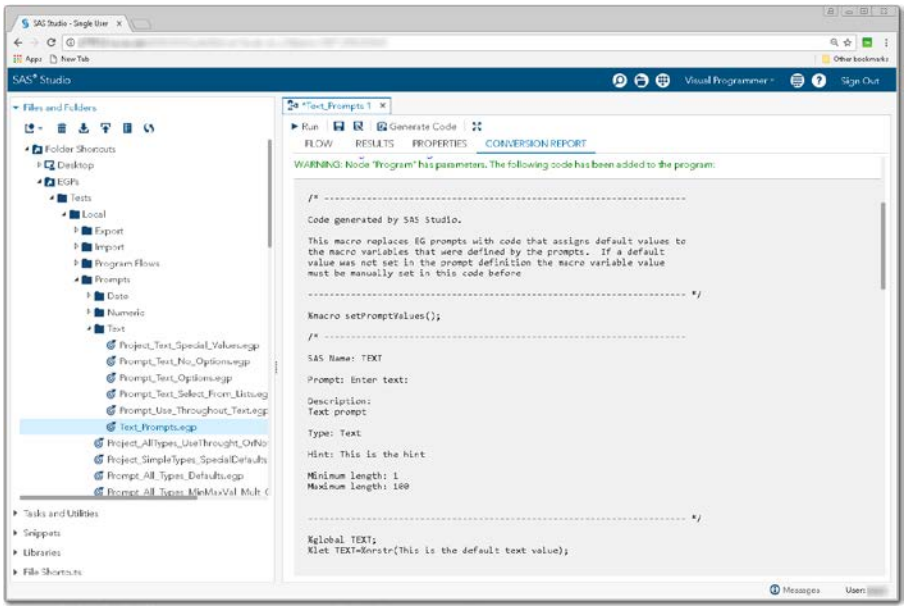


Display 94 - %SYMDEL Statement Removes Text Macro Variable

SAS Studio

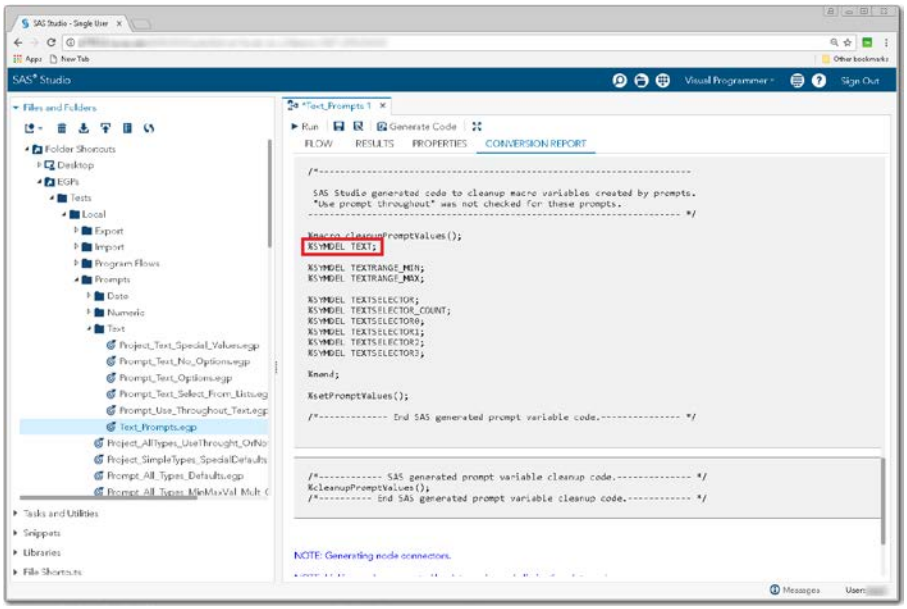
The following display shows the code that is added to the converted Program node for the text prompt in SAS Enterprise Guide.

A global variable named TEXT is created and a %LET statement assigns the default value to TEXT. If you want to run the process flow using a different value for the TEXT prompt, you must manually update the value of the macro variable in the %LET statement.



Display 95 - Generated Macro Code for Single Value Text Prompt

Because the **Use prompt value throughout project** option is not checked, the %SYMDEL statement removes the TEXT macro variable.

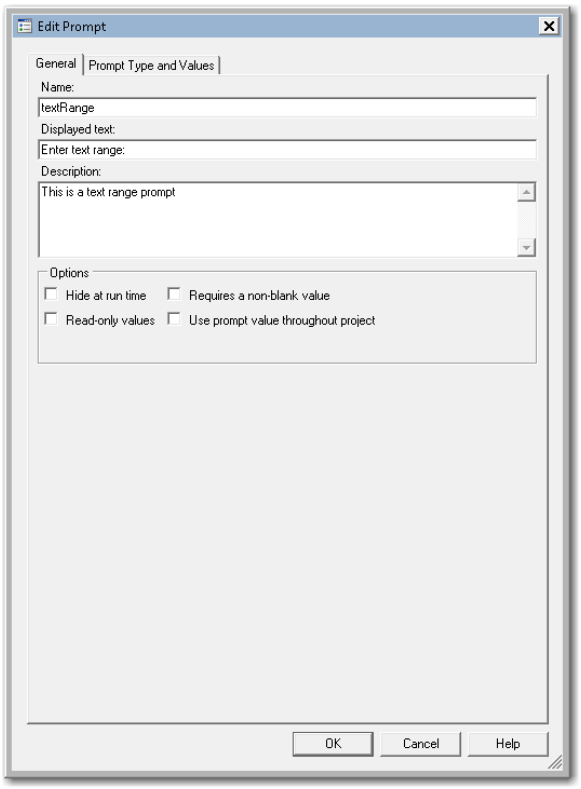


Display 96 - %SYMDEL Statement Removes TEXT Macro Variable

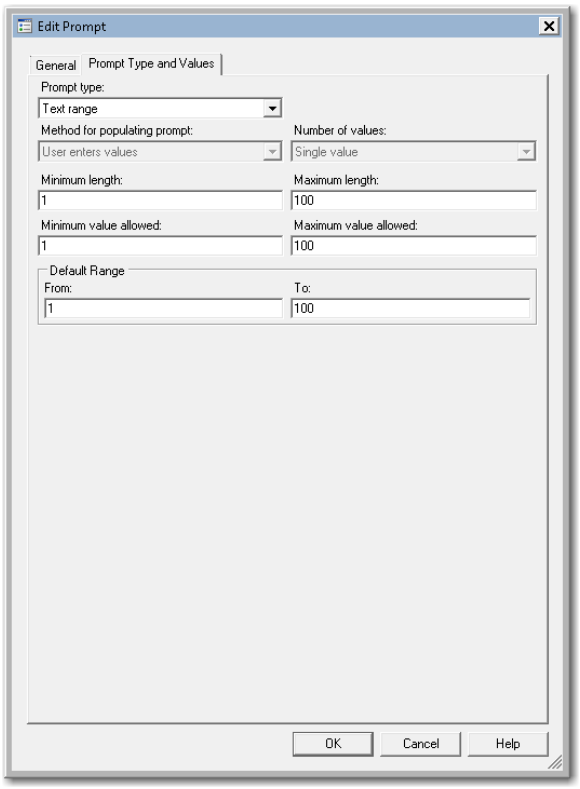
Text Range

SAS Enterprise Guide

In this example, a text range prompt named textRange is defined as shown in the following two displays.

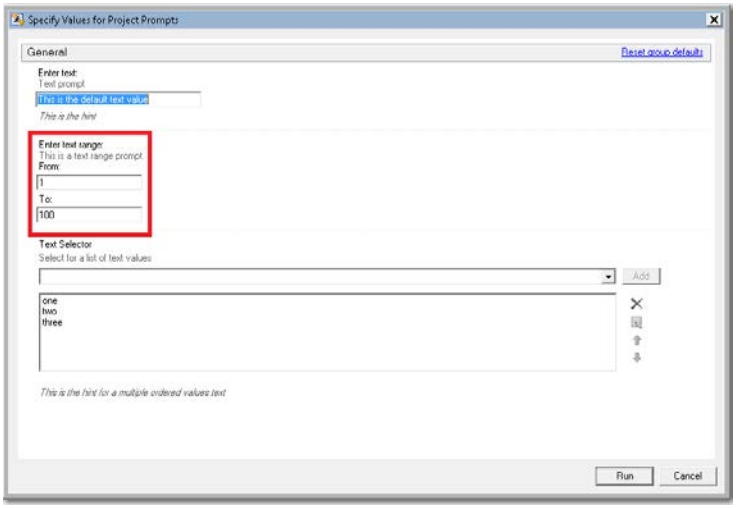


Display 97 - General Properties for Text Range Prompt



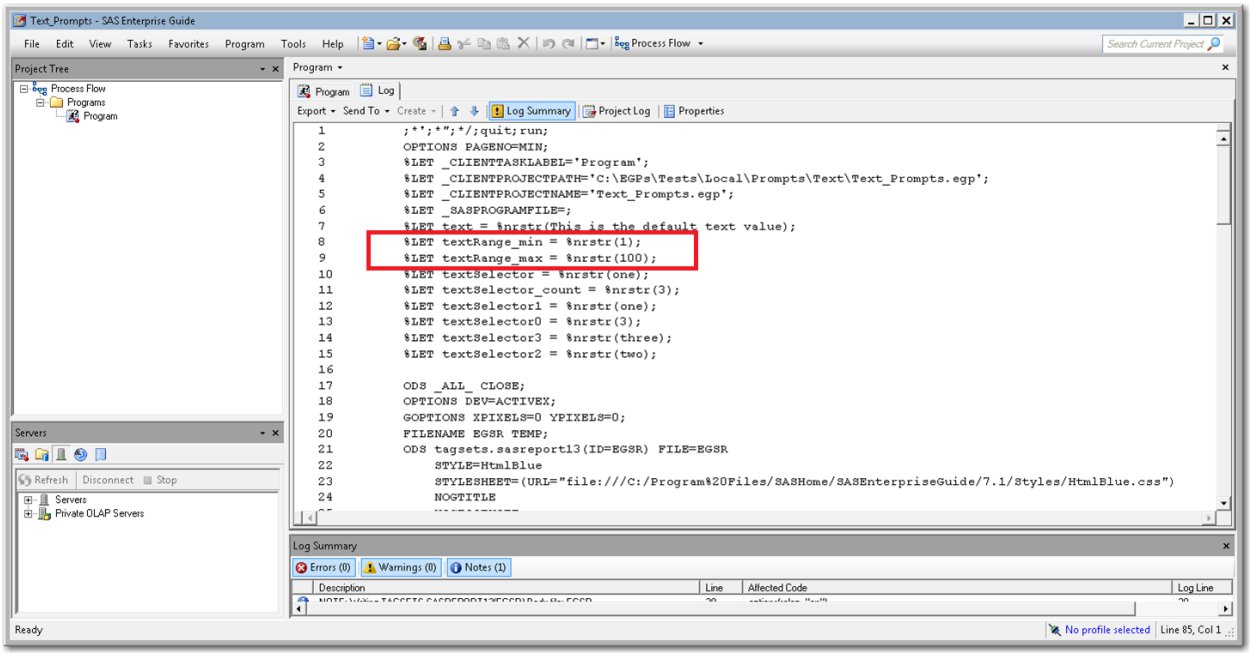
Display 98 - Type and Values for Text Range Prompt

When you run the Program node that depends on this prompt, the following dialog box appears.



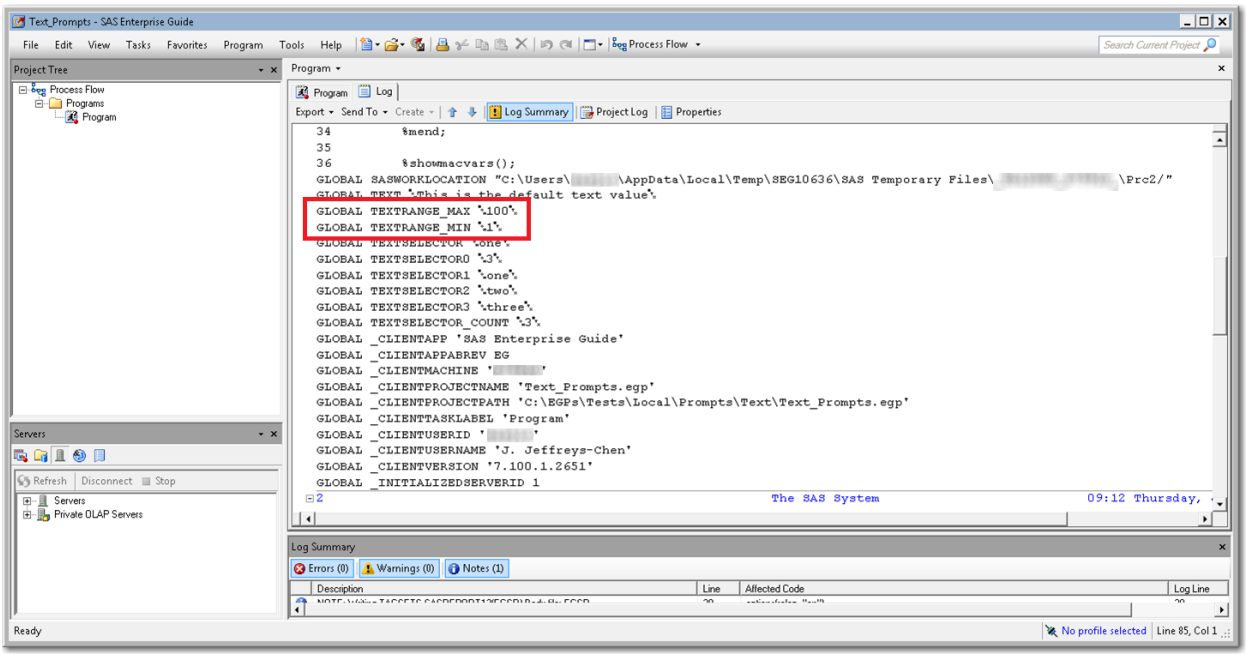
Display 99 - Text Range Prompt in Prompt Dialog Box

If the user leaves the default values in the text range prompt fields, the following code is generated by SAS Enterprise Guide in the Program node that depends on the prompt. The %LET statements assign the values specified in the prompt dialog box to the textRange_min and textRange_max macro variables.



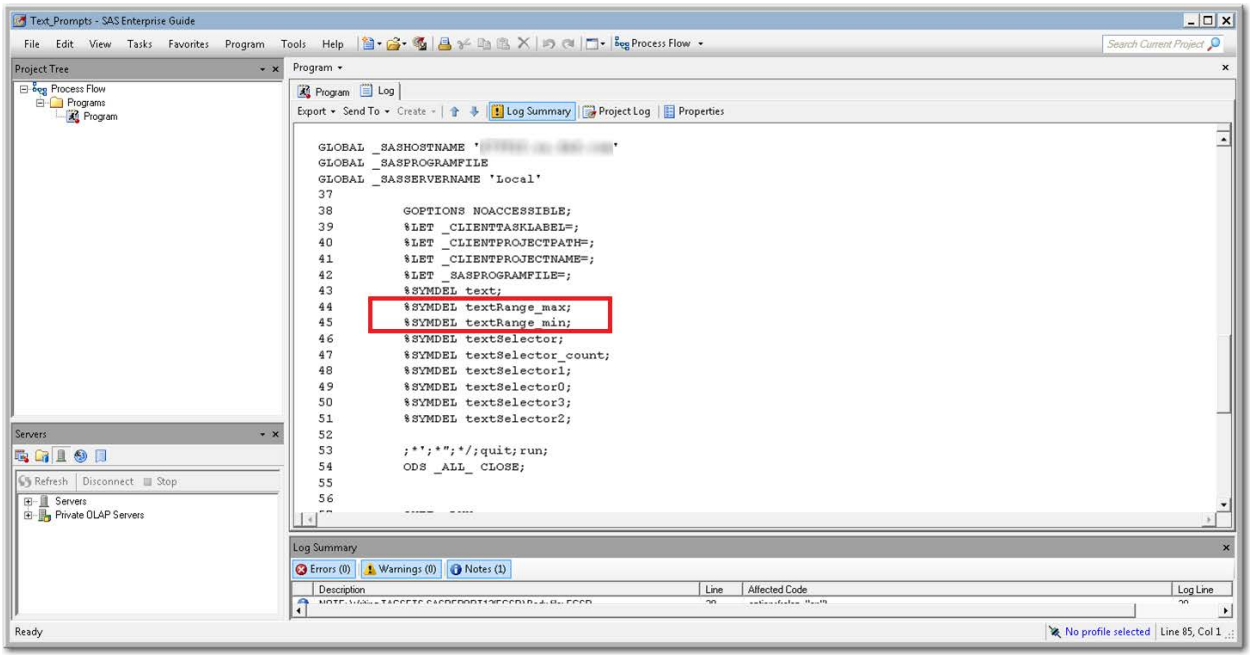
Display 100 - %LET Statements for textRange_min and textRange_max Macro Variables

The log of the [Program node using the prompt definition] displays the value of the global variables created by the prompt.



Display 101 – Values for Global Variables for Text Range Prompt

Because the **Use prompt value throughout project** option is not checked for this prompt, the %SYMDEL statements remove the macro variables at the end of the program.

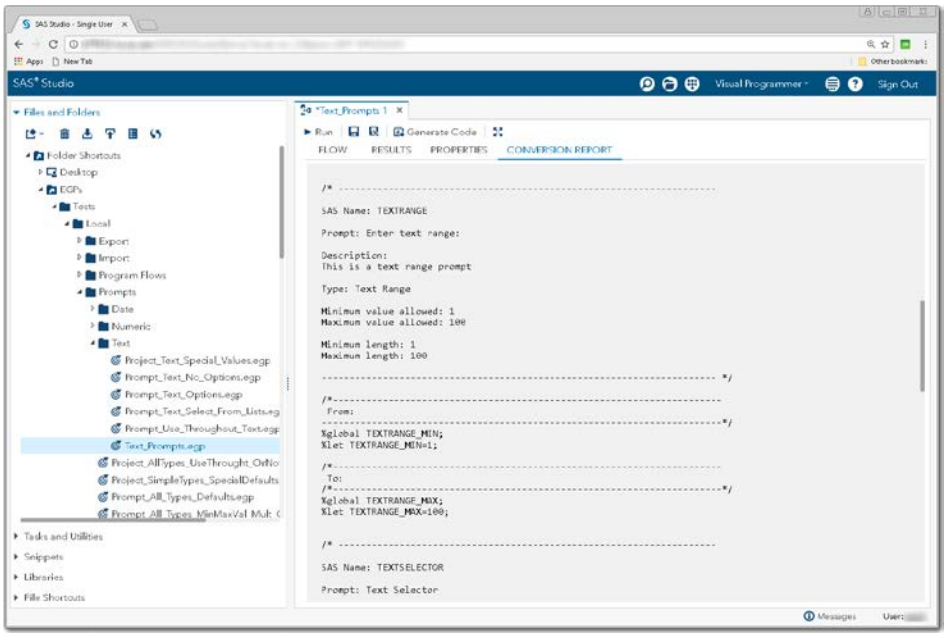


Display 102 - %SYMDEL Statements Remove the textRange_max and textRange_min Macro Variables

SAS Studio

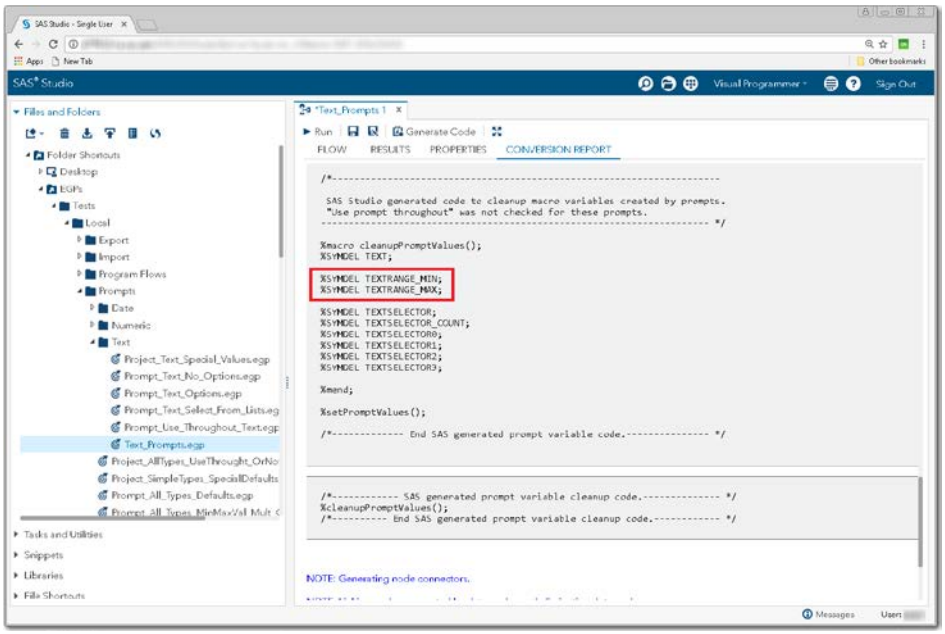
The following display shows the code that is added to the converted Program node for the text range prompt in SAS Enterprise Guide.

Global variables named TEXTRANGE_MIN and TEXTRANGE_MAX are created. The %LET statements assign the default values to the TEXTRANGE_MIN and TEXTRANGE_MAX variables. If you want to run your process flow using different values for the TEXTRANGE prompt, you must manually update the values of the macro variables in the %LET statements.



Display 103 - Generated Macro Code for Text Range Prompt

Because the **Use prompt value throughout project** option is not checked, the %SYMDEL statements remove the TEXTRANGE* macro variables.

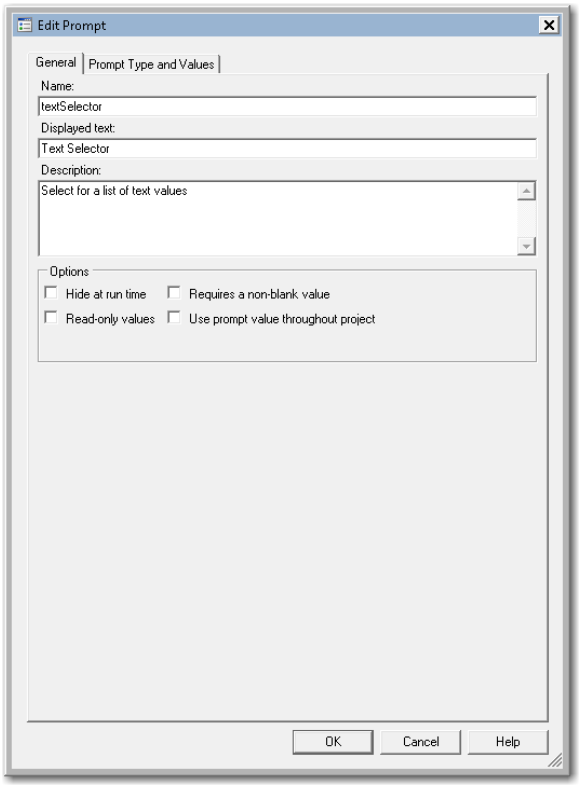


Display 104 - %SYMDEL Statements Remove the TEXTRANGE* Macro Variables

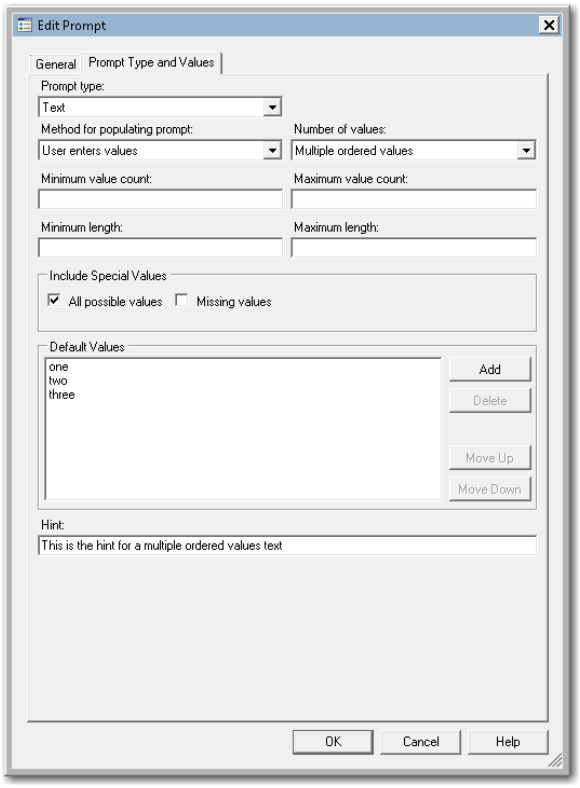
Multiple Text Values

SAS Enterprise Guide

In this example, a multiple ordered values text prompt named textSelector is defined as shown in the following two displays.

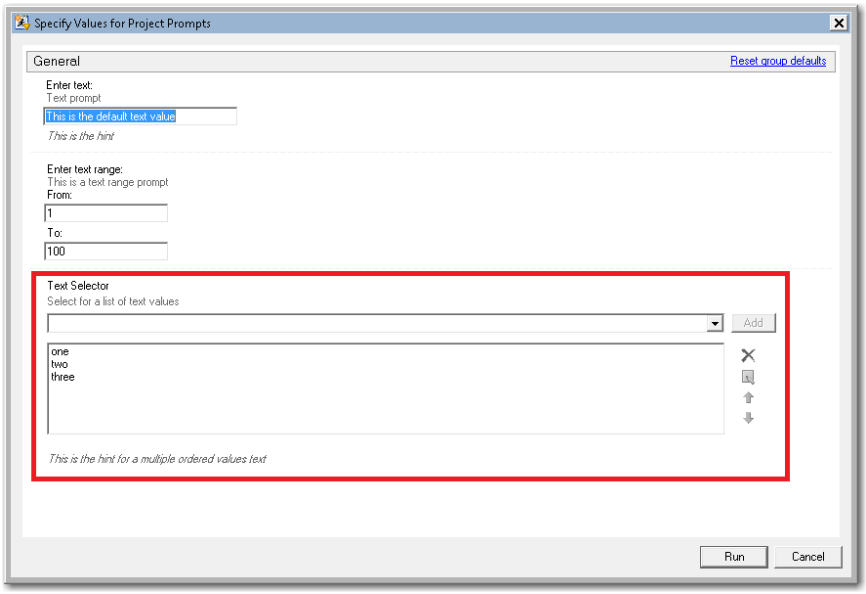


Display 105 - General Properties for Multiple Text Values Prompt



Display 106 - Type and Values for Multiple Text Values Prompt

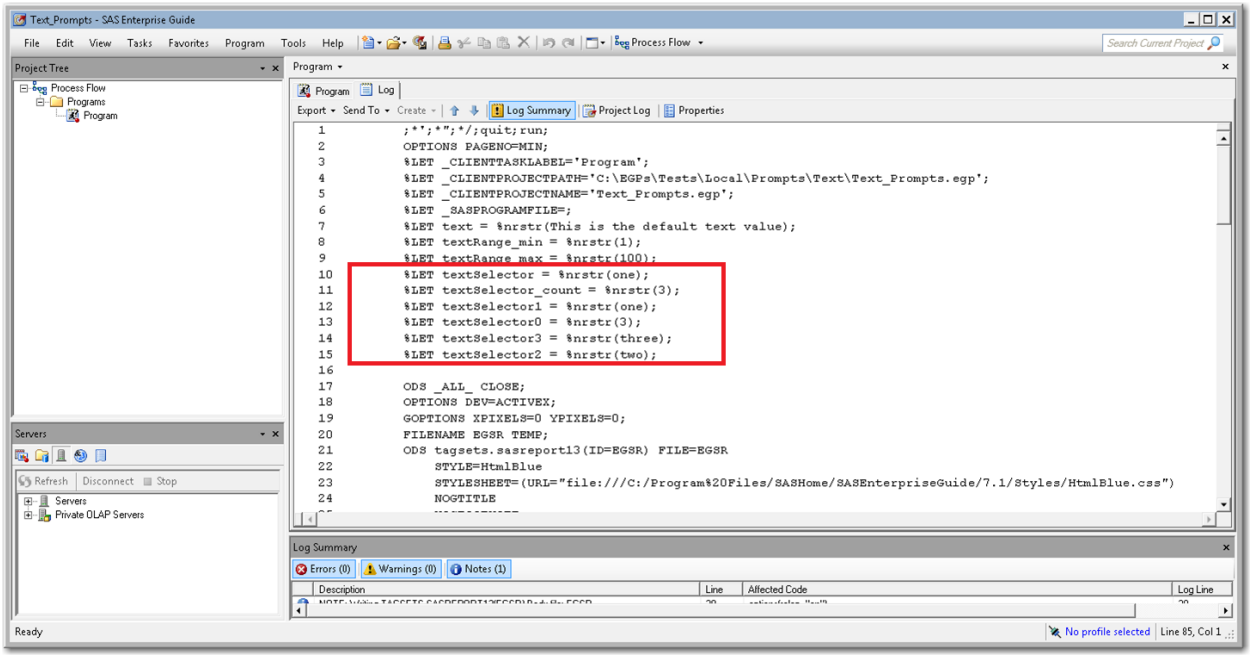
When you run the Program node that depends on this prompt, the following dialog box appears.



Display 107 - Multiple Text Values Prompt in Prompt Dialog Box

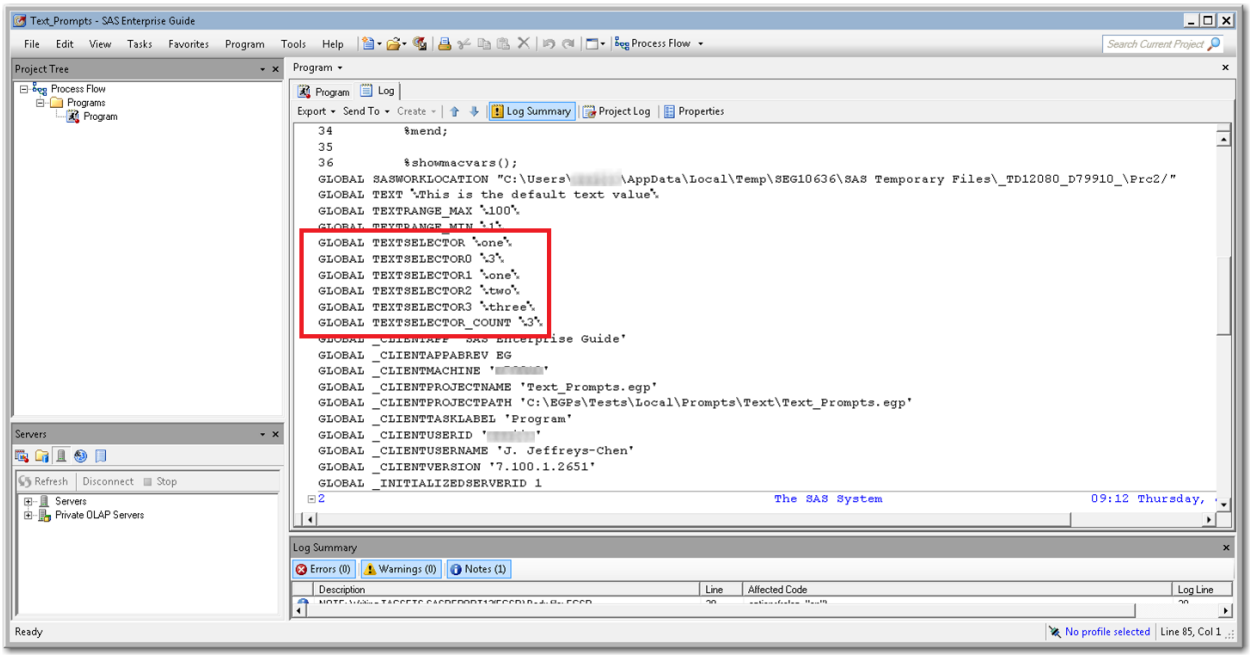
If the user leaves the default values for the multiple text values prompt, the textSelector* macro variables are generated by SAS Enterprise Guide in the Program node that depends on the prompt.

The %LET statements assign the values specified in the prompt dialog box to the textSelector* macro variables.



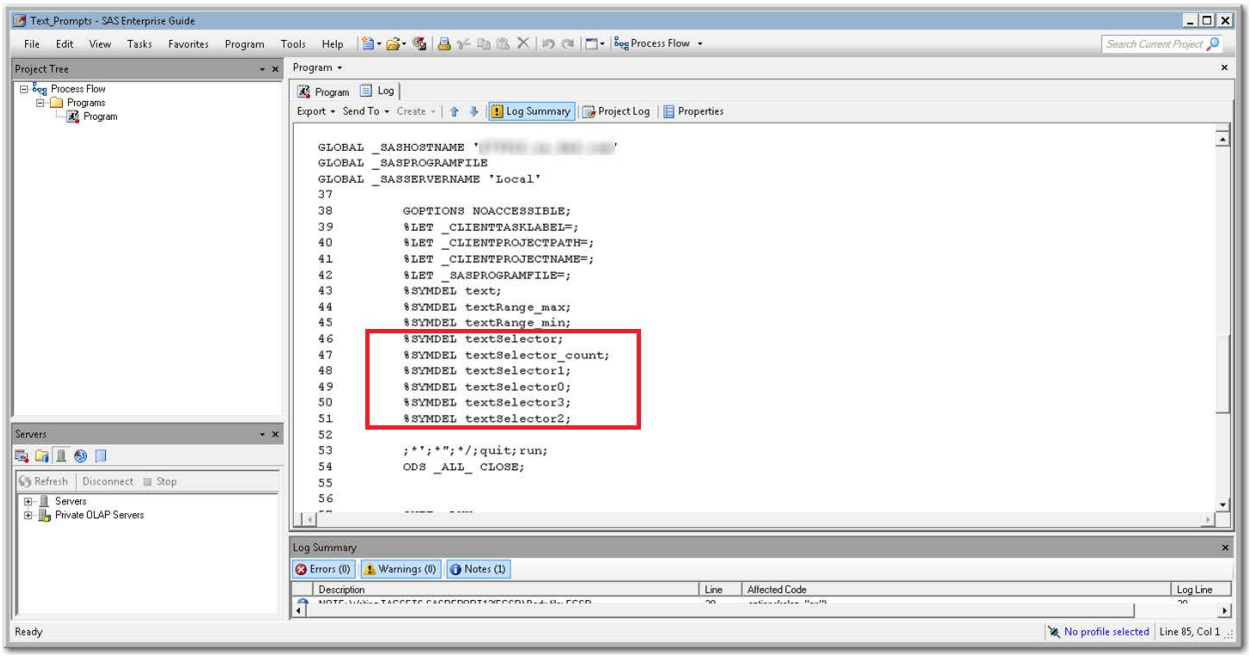
Display 108 - %LET Statements for Multiple Value Text Prompt

The log of the [Program node using the prompt definition] displays the values of the global variables created by the prompt.



Display 109 – Global Variables for Multiple Value Text Prompt

Because the **Use prompt value throughout project** option is not checked for this prompt, the %SYMDEL statements remove the macro variables at the end of the program.



Display 110 - %SYMDEL Statements Remove the textSelector* Macro Variables

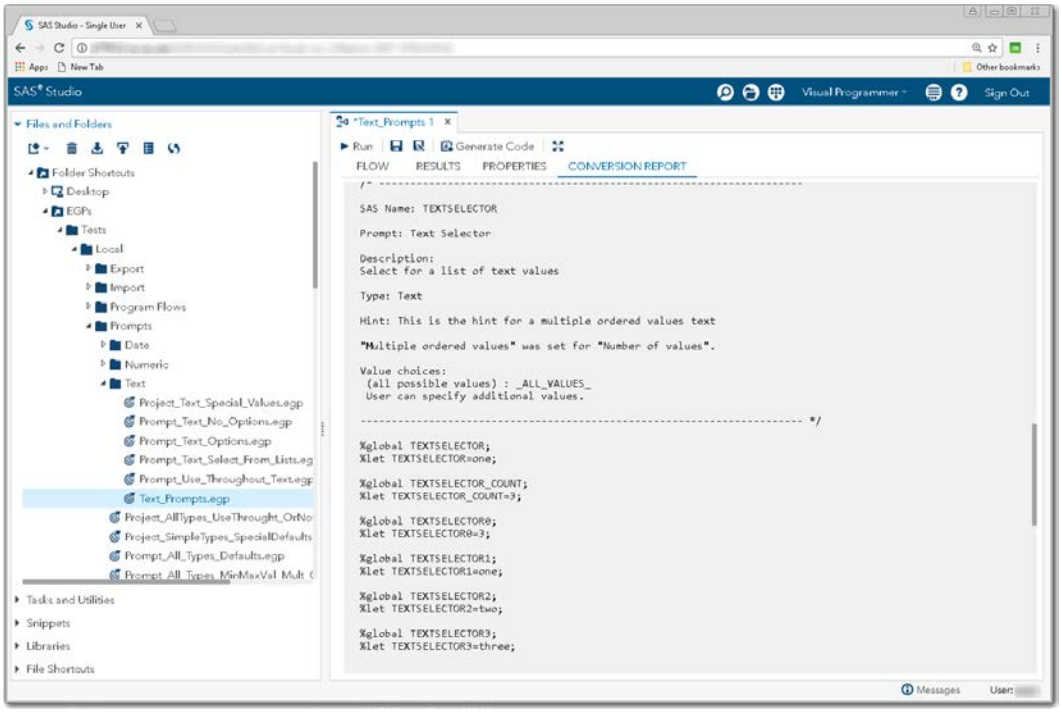
SAS Studio

The following display shows code that is added to the converted Program node for the multiple text values prompt in SAS Enterprise Guide. These global variables are created:

- TEXTSELECTOR
- TEXTSELECTOR_COUNT
- TEXTSELECTOR0
- TEXTSELECTOR1
- TEXTSELECTOR2
- TEXTSELECTOR3

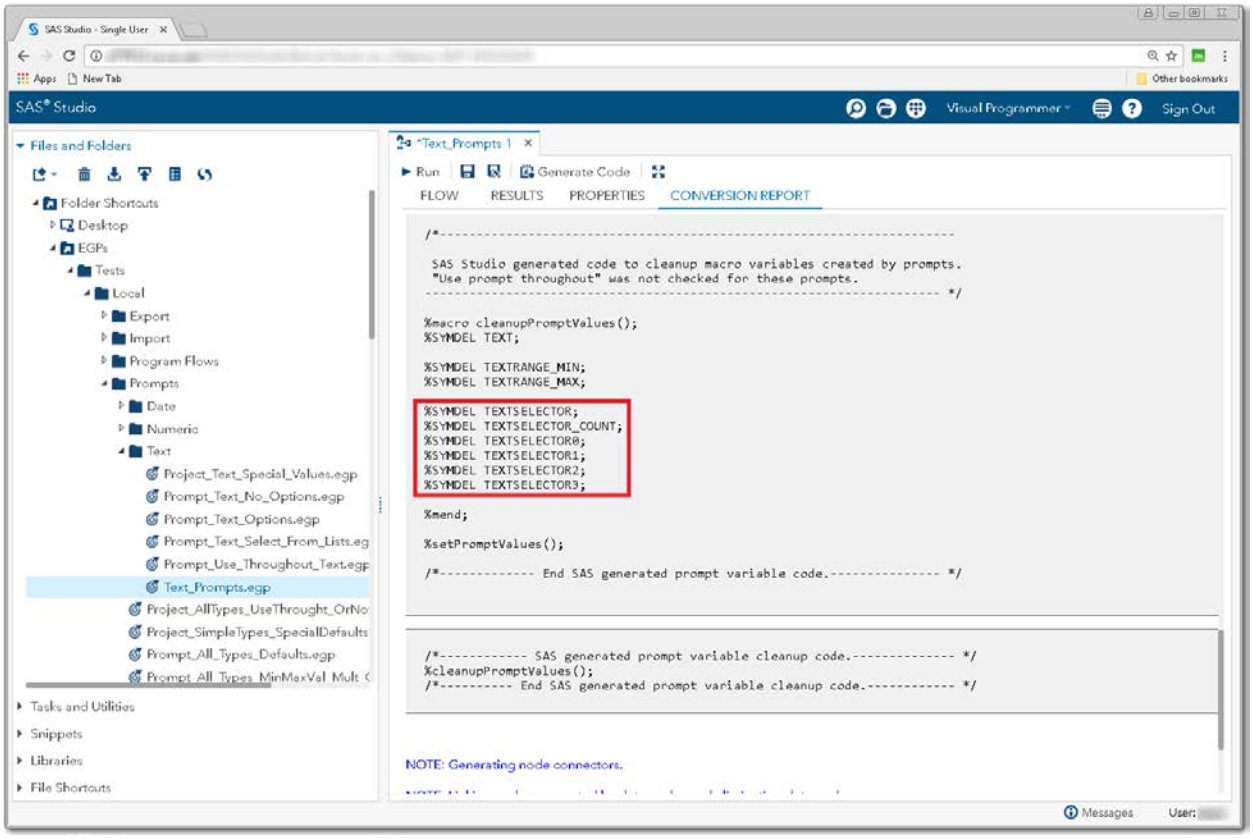
The %LET statements assign the default values to the TEXTSELECTOR1, TEXTSELECTOR2, and TEXTSELECTOR3 variables.

If you want to run your process flow using different values for the TEXTSELECTOR prompt, you must manually update values of the macro variables values in the %LET statements. Note that in this example, the TEXTSELECTOR_COUNT and TEXTSELECTOR_0 variables must reflect the number of text selections you would like your program to process, and the TEXTSELECTORn variables must be in sequential order.



Display 111 - Macro Code for Multiple Text Values Prompt

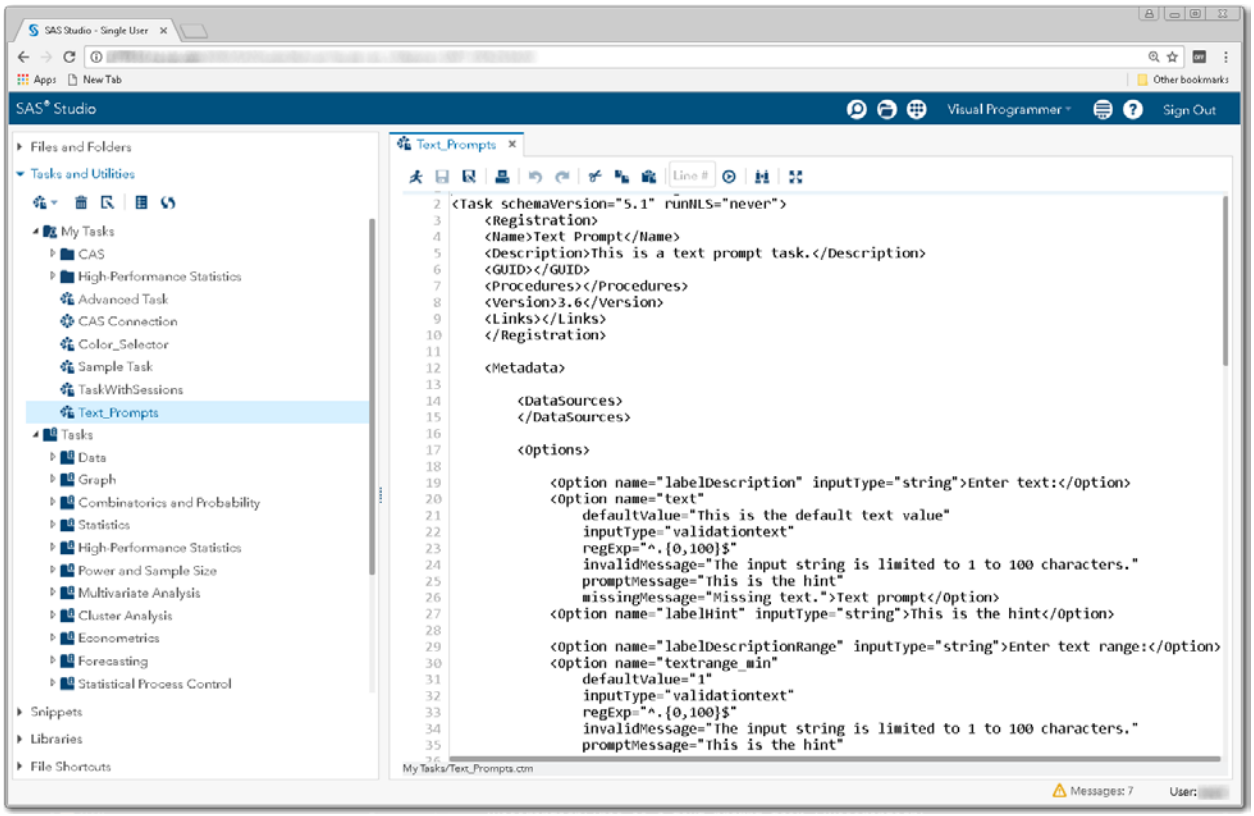
Because the **Use prompt value throughout project** option is not checked, the %SYMDEL statements remove the TEXTSELECTOR* macro variables.



Display 112 - %SYMDEL Statements Remove TEXTSELECTOR* Macro Variables

Substituting a SAS Studio Task for Text Prompts

1. Create a SAS Studio task with controls that represents the text prompts.
   - Add a label and text input controls. The names of the input controls should match the prompt names of text, textRange_min, textRange_max and so on.
   - Set the default values to the default values shown in the generated `setPromptValues()` macro in the converted Program node.
   - Change the strings of the input controls to match the strings specified in the prompts.



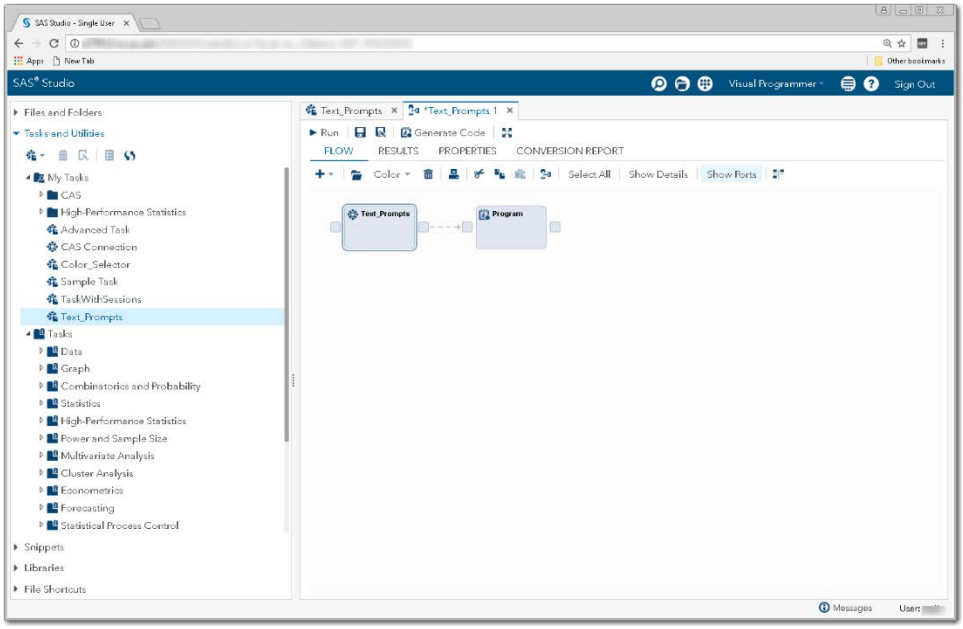
Display 113 - Replacement Task for Multiple Text Values Prompt

The following code is an example of a task that could be used for text prompts.

```
<?xml version="1.0" encoding="UTF-8"?>
<Task schemaVersion="5.1" runNLS="never">
       <Registration>
       <Name>Text Prompt</Name>
       <Description>This is a text prompt task.</Description>
       <GUID></GUID>
       <Procedures></Procedures>
       <Version>3.6</Version>
       <Links></Links>
       </Registration>
       <Metadata>

              <DataSources>
              </DataSources>

              <Options>

                     <Option name="labelDescription" inputType="string">
                         Enter text:
                     </Option>
                     <Option name="text"
                           defaultValue="This is the default text value"
                           inputType="validationtext"
                           regExp="^.{0,100}$"
                           invalidMessage=
                               "The input string is limited to 1 to 100 characters."
                           promptMessage="This is the hint"
                           missingMessage="Missing text.">
                         Text prompt
                     </Option>


                     <Option name="labelHint" inputType="string">
                         This is the hint
                     </Option>
                     <Option name="labelDescriptionRange" inputType="string">
                         Enter text range:
                     </Option>
                     <Option name="textrange_min"
                           defaultValue="1"
                           inputType="validationtext"
                           regExp="^.{0,100}$"
                           invalidMessage=
                               "The input string is limited to 1 to 100 characters."
                           promptMessage="This is the hint"
                           missingMessage="Missing text.">
                         From:
                     </Option>
                     <Option name="textrange_max"
                           defaultValue="100"
                           inputType="validationtext"
                           regExp="^.{0,100}$"
                           invalidMessage=
                               "The input string is limited to 1 to 100 characters."
                           promptMessage="This is the hint"
                           missingMessage="Missing text.">
                         to:
                     </Option>
```

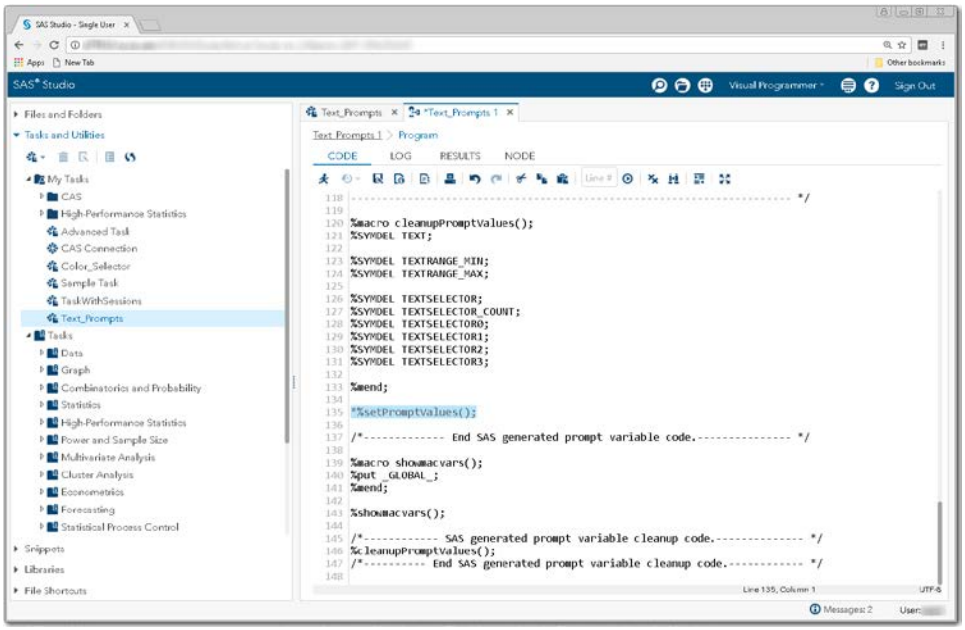
```
                    <Option name="labelDescriptionTextSel" inputType="string">
                        Text selector
                    </Option>
                    <Option name="multientry" inputType="multientry">
                           Select for a list of text values
                    </Option>
                    <Option name="one" inputType="string">one</Option>
                    <Option name="two" inputType="string">two</Option>
                    <Option name="three" inputType="string">three</Option>
                    <Option name="labelDescriptionTextSelHint" inputType="string">
                         This is the hint for a multiple ordered values text
                    </Option>
             </Options>
      </Metadata>
      <UI>
             <OptionItem option="labelDescription"/>
             <OptionItem option="text"/>
             <OptionItem option="labelHint"/>

             <OptionItem option="labelDescriptionRange"/>
             <OptionItem option="textrange_min"/>
             <OptionItem option="textrange_max"/>

             <OptionItem option="labelDescriptionTextSel"/>

            <OptionChoice option="multientry">
                    <OptionItem option="one"/>
                    <OptionItem option="two"/>
                    <OptionItem option="three"/>
             </OptionChoice>
             <OptionItem option="labelDescriptionTextSelHint"/>
      </UI>
      <CodeTemplate>
             <![CDATA[
%global text;
%let text=$text;

%global textrange_min;
%global textrange_max;
%let textrange_min=$textrange_min;
%let textrange_max=$textrange_max;

%global multientry;
%let multientry=$multientry;

#if ($multientry.size()>0)
%global textselector_count;
%let textselector_count=$multientry.size();
%global textselector0;
%let textselector0=$multientry.size();
%global textselector;
%let textselector=$multientry[0];

#set($counter = 1)
#foreach ($id in $multientry)
   %global textselector$counter;
   %let textselector$counter=$id;
   #set($counter = $counter + 1)
#end
#end
             ]]>
      </CodeTemplate>
</Task>
```

2. Save the prompt replacement task to your **My Tasks** folder.
3. Drag the task from **My Tasks** into your converted process flow.
4. Link the output port of the Task node to the input port for the converted Program node.



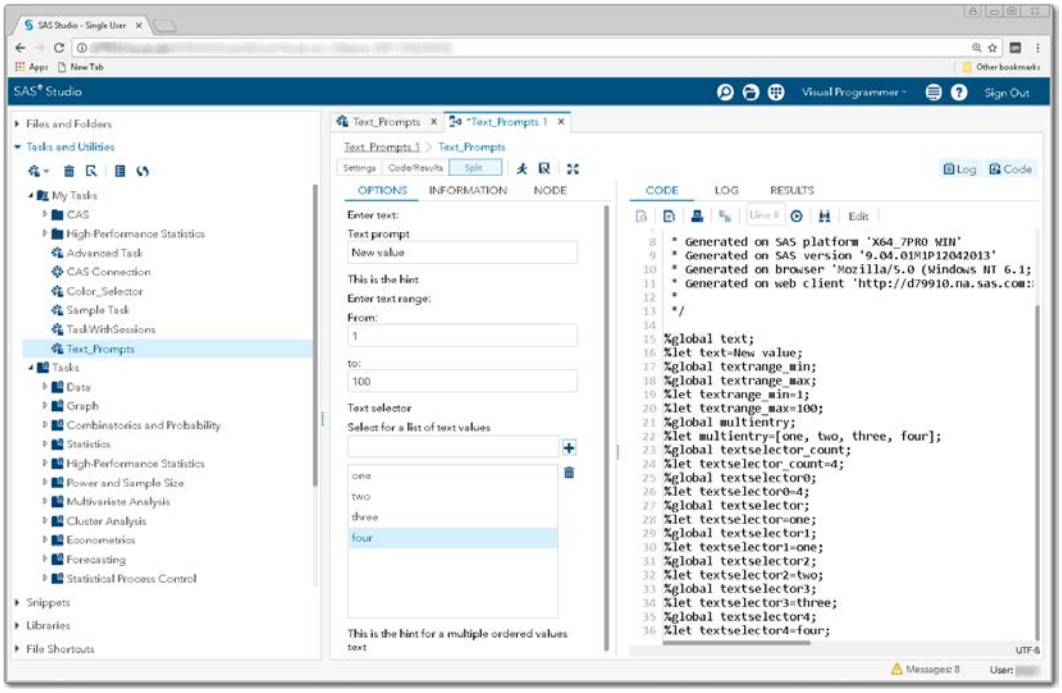
Display 114 – Linking the Output from the Task Node to the Program Node

5. Comment out the `%setPromptValues()` macro call in the converted Program node. The macro code generated by the new task replaces this code.



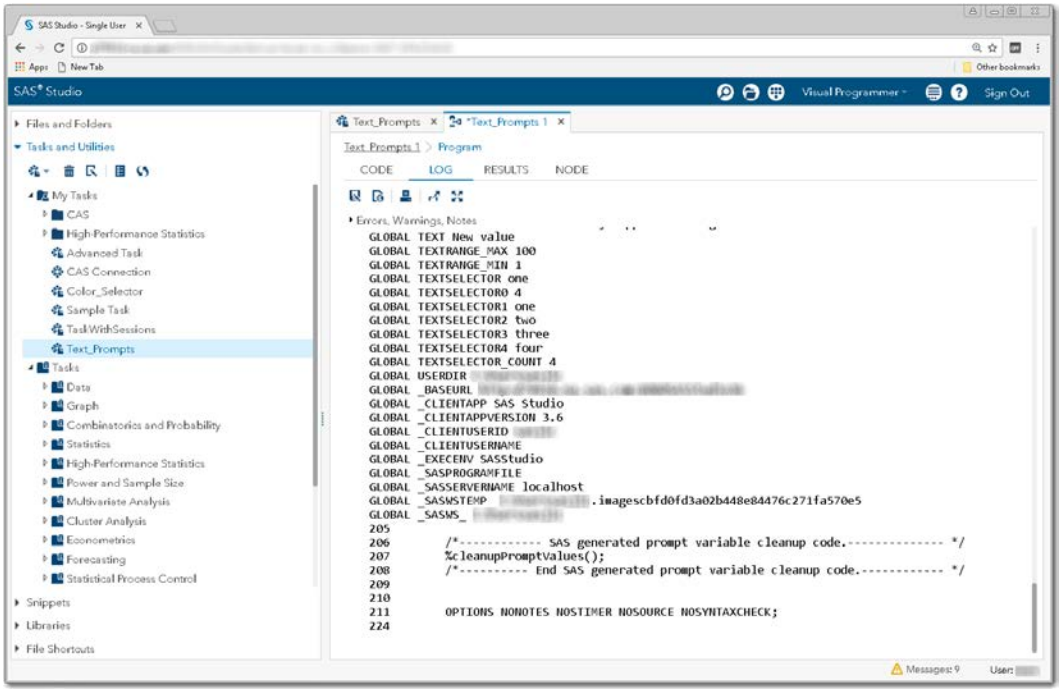
Display 115 - Commented Out %setPromptValues() Macro Call

To run your flow with text values other than the default values, open the Text_Prompts node and specify different values.



Display 116 – User Interface and Generated SAS Code for Text_Prompts Task

When you execute the process flow, the global Text* variables are set to the values that you specified in the new task.



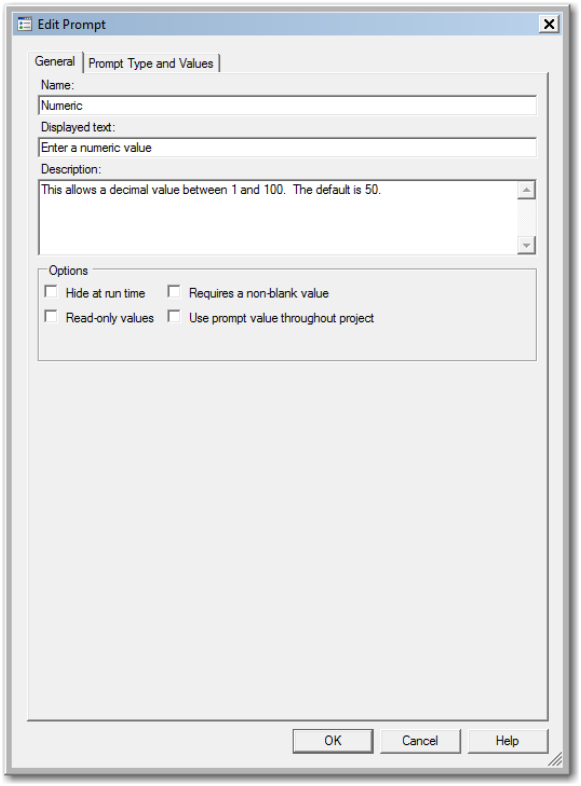
Display 117 - Text Prompt Variables with Updated Values

Numeric

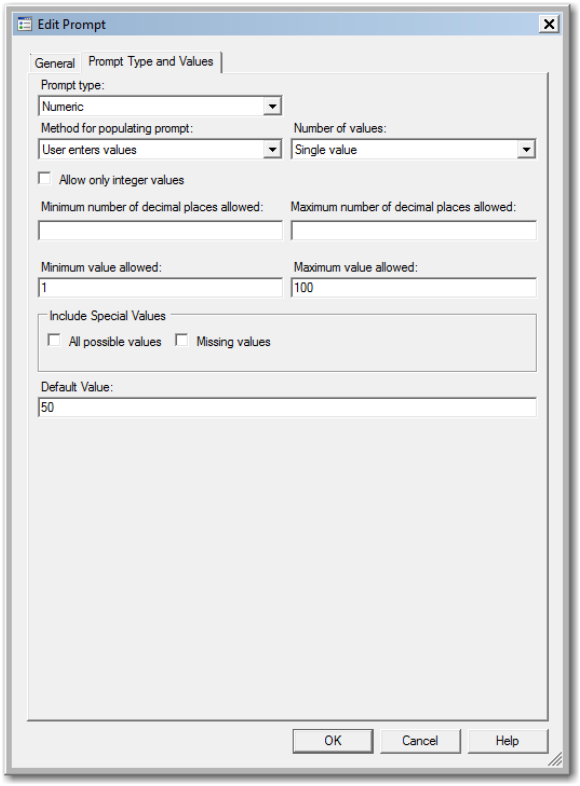
Single Number

SAS Enterprise Guide

In this example, a single value numeric prompt named Numeric is defined as shown in the following two displays.

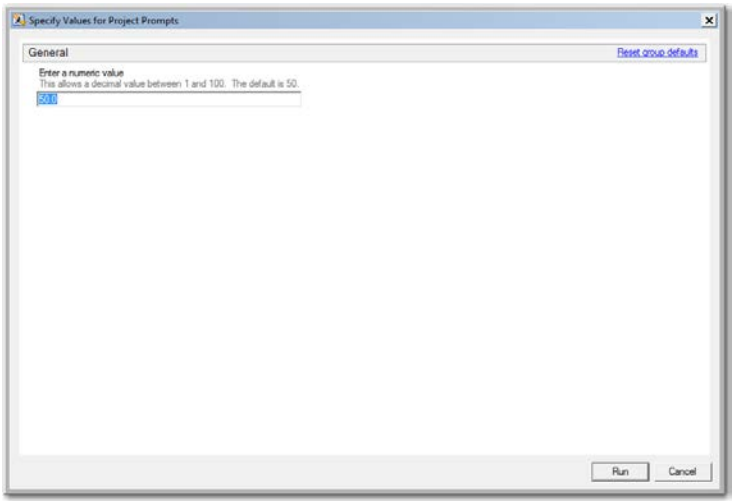


Display 118 - General Properties for Numeric Prompt



Display 119 - Type and Values for Numeric Prompt

When you run the Program node that depends on this prompt, the following dialog box appears:

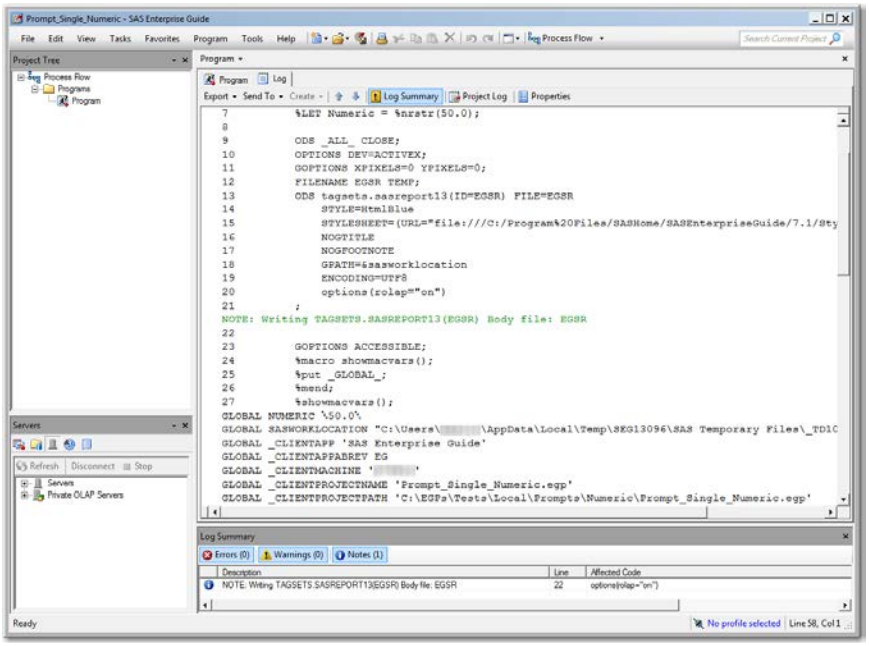


Display 120 - Numeric Prompt in Prompt Dialog Box

If the user leaves the default value in the single numeric value prompt field, the following code is generated by SAS Enterprise Guide in the Program node that depends on the prompt.

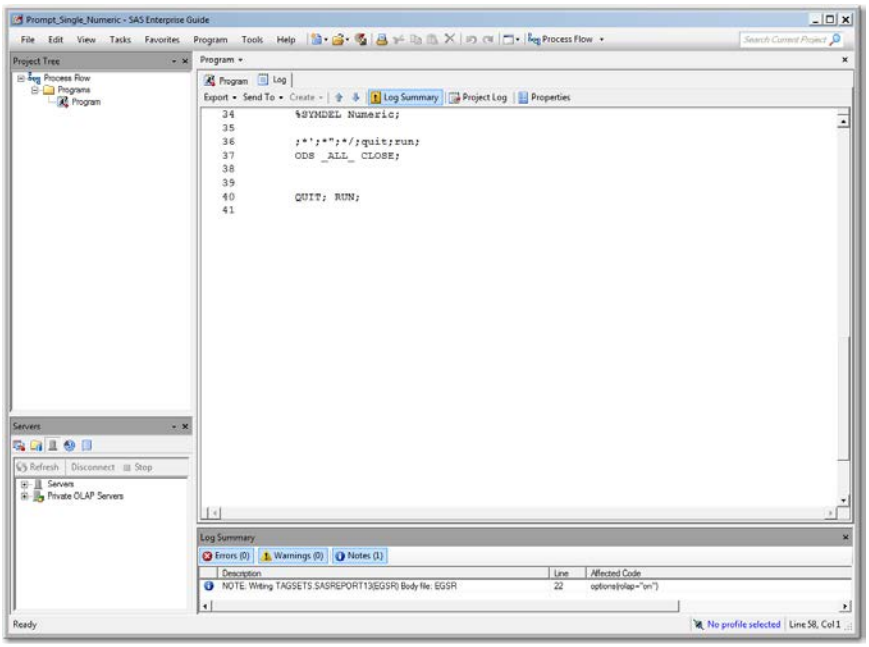
A %LET statement assigns the value specified in the prompt dialog box to the Numeric macro variable.

The log of the Program node using the prompt definition displays the value of the global variable created by the prompt.



Display 121 - Global Variable and %LET Statement for Numeric Prompt

Because the **Use prompt value throughout project** option is not checked for this prompt, the %SYMDEL statement removes the macro variable at the end of the program.

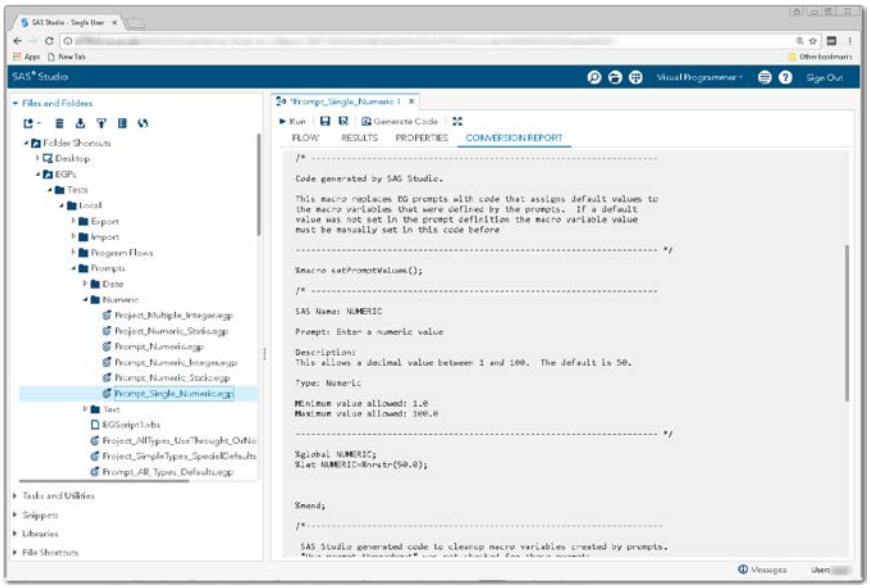


Display 122 - %SYMDEL Statement Removes Numeric Macro Variable

SAS Studio

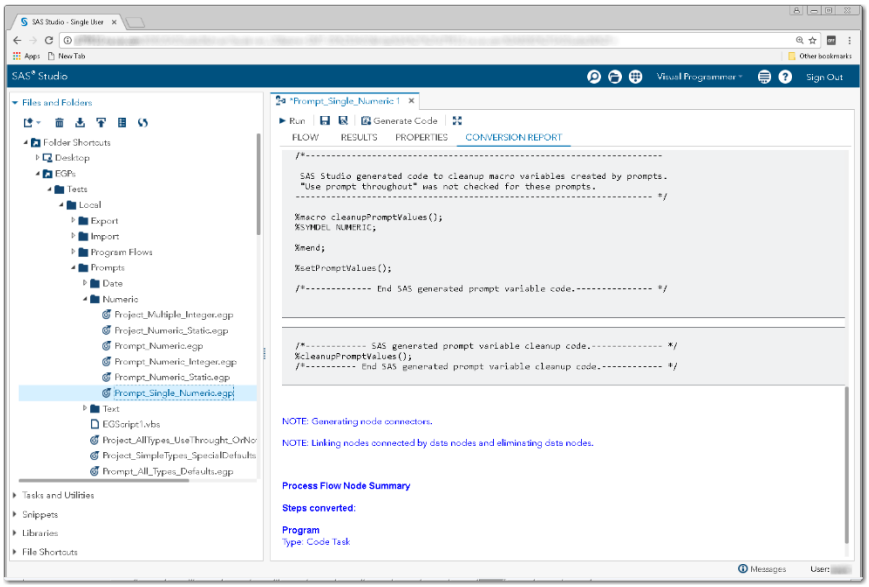
The following display shows code that is added to the converted Program node for the numeric prompt in SAS Enterprise Guide.

A global variable named NUMERIC is created and a %LET statement assigns the default value to NUMERIC**.** If you want to run your process flow using different values for the NUMERIC prompt, you must manually update value of the macro variable in the %LET statement.



Display 123 - Macro Code for Numeric Prompt

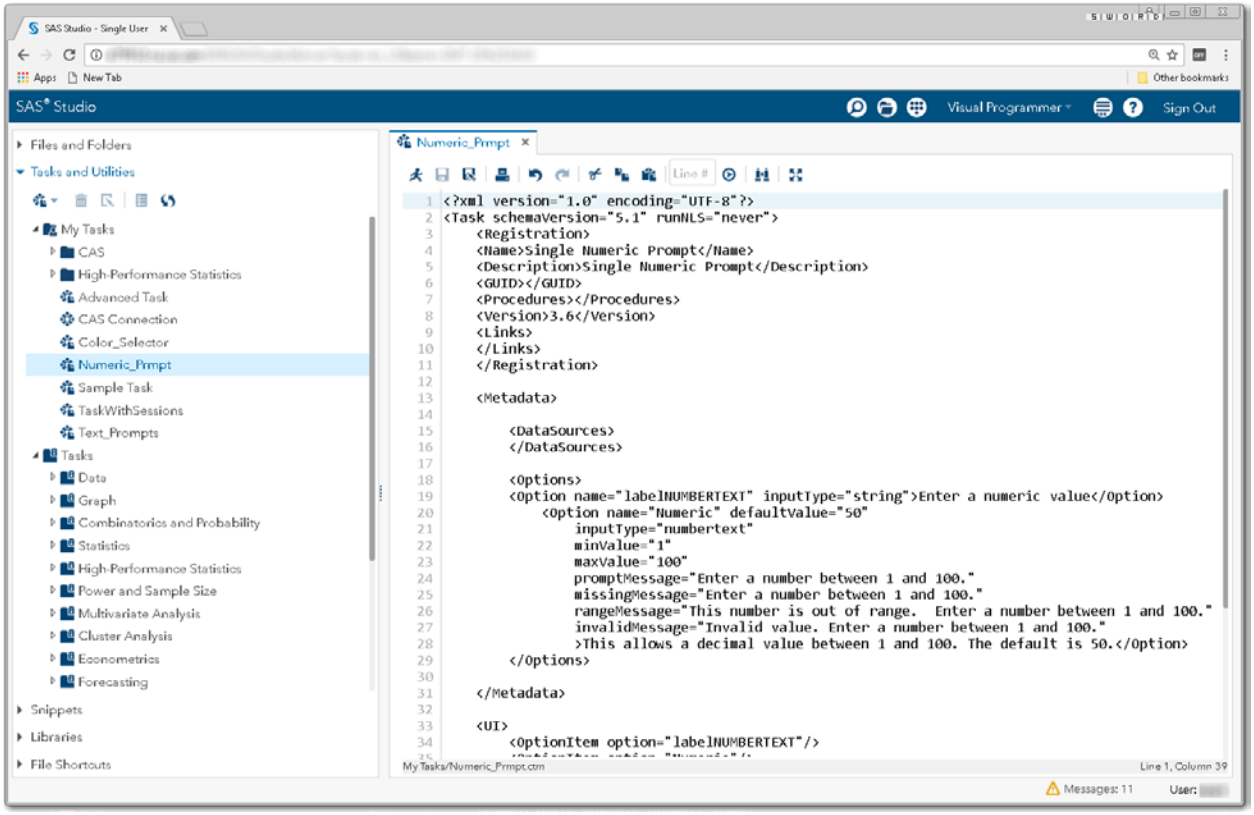
Because the **Use prompt value throughout project** option is not checked, the %SYMDEL statements remove the NUMERIC macro variable.



Display 124 - %SYMDEL Statements Remove Numeric Macro Variable

Substituting a SAS Studio Task for Numeric Prompt

1. Create a SAS Studio task with a control that represents the Numeric prompt.
   - Add a label and a numbertext input control. The name of the input control should match the prompt name of Numeric.
   - Set the default value to the default value shown in the generated `setPromptValues()` macro in the converted Program node.
   - Change the prompt string to match the string specified in the prompt.

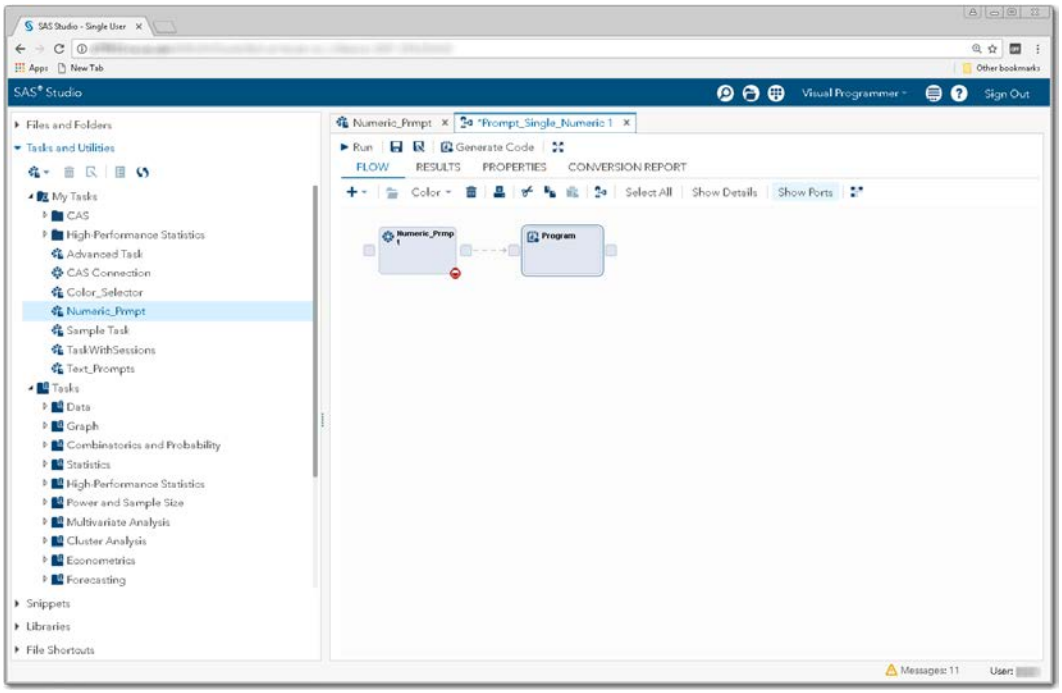


Display 125 - Replacement Task for Numeric Prompt

The following code is an example of a task that could be used as the numeric prompt.

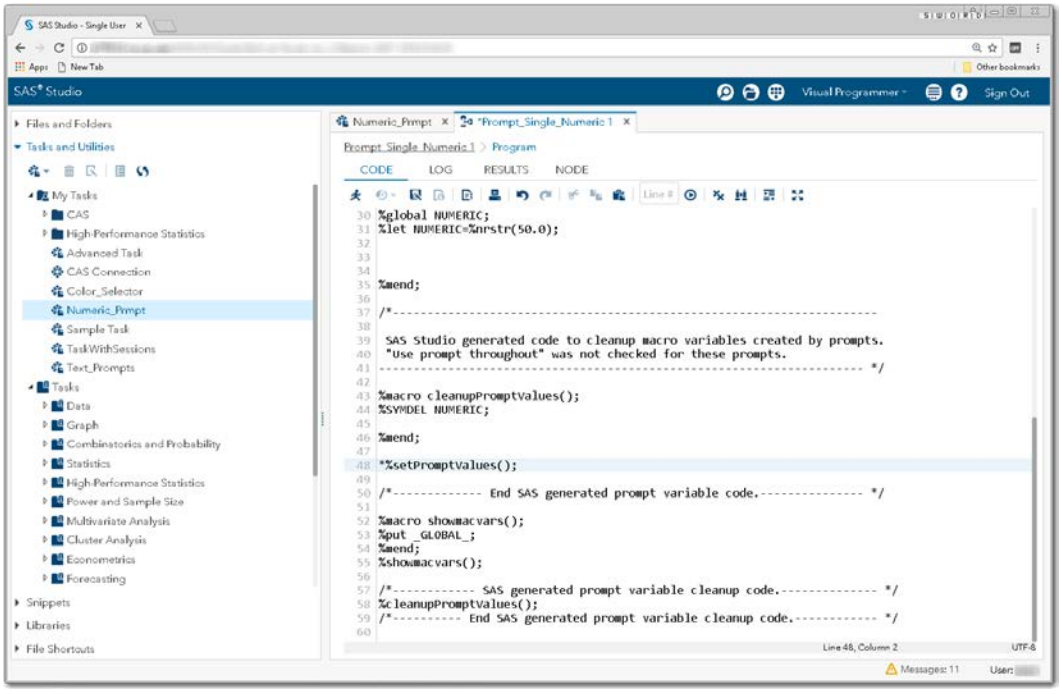
```
<?xml version="1.0" encoding="UTF-8"?>
<Task schemaVersion="5.1" runNLS="never">
       <Registration>
       <Name>Single Numeric Prompt</Name>
       <Description>Single Numeric Prompt</Description>
       <GUID></GUID>
       <Procedures></Procedures>
       <Version>3.6</Version>
       <Links>
       </Links>
       </Registration>

       <Metadata>

              <DataSources>
              </DataSources>

              <Options>
                     <Option name="labelNUMBERTEXT" inputType="string">
                         Enter a numeric value
                     </Option>

                      <Option name="Numeric" defaultValue="50"
                           inputType="numbertext"
                           minValue="1"
                           maxValue="100"
                           promptMessage="Enter a number between 1 and 100."
                           missingMessage="Enter a number between 1 and 100."
                           rangeMessage=
                      "This number is out of range.  Enter a number between 1 and 100."
                           invalidMessage=
                             "Invalid value. Enter a number between 1 and 100."
                           >
                      This allows a decimal value between 1 and 100. The default is 50.
                       </Option>
              </Options>

       </Metadata>

       <UI>
              <OptionItem option="labelNUMBERTEXT"/>
              <OptionItem option="Numeric"/>
       </UI>

       <CodeTemplate>
              <![CDATA[

%global Numeric;
%let Numeric=$Numeric;


              ]]>
       </CodeTemplate>
</Task>
```

2. Save the prompt replacement task to your **My Tasks** folder.
3. Drag the task from **My Tasks** into your converted process flow.
4. Link the output port of the new task to the input port of the converted Program node.



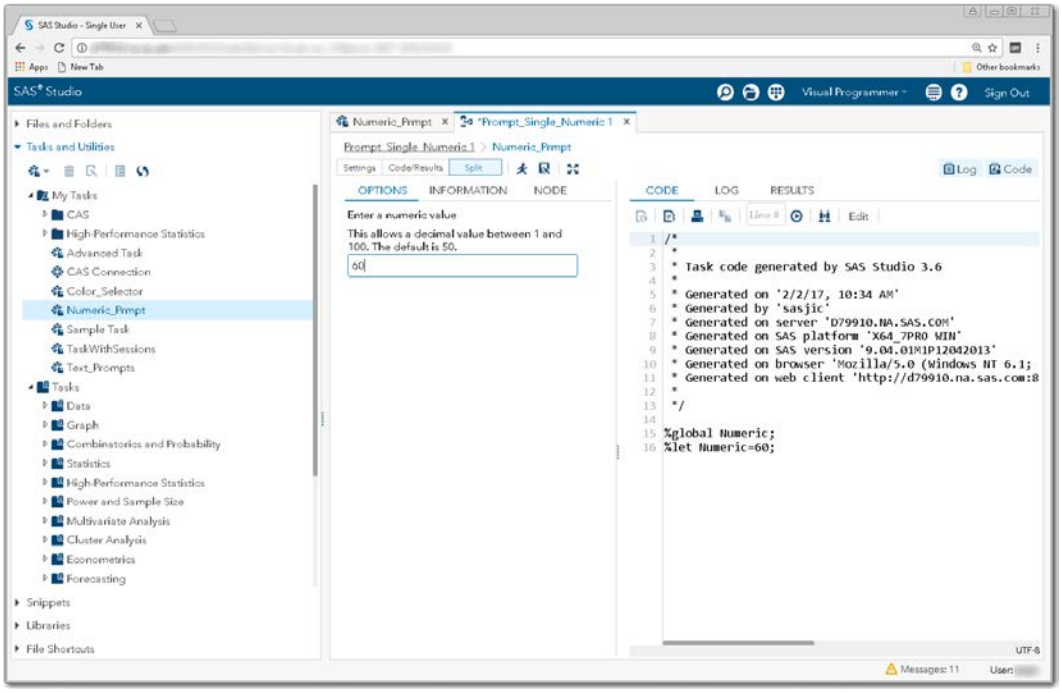
Display 126 - Numeric Input Task Linked to Program Node

5. Comment out the `%setPromptValues()` macro call from the converted Program node. The macro code generated by the numeric input task replaces this code.



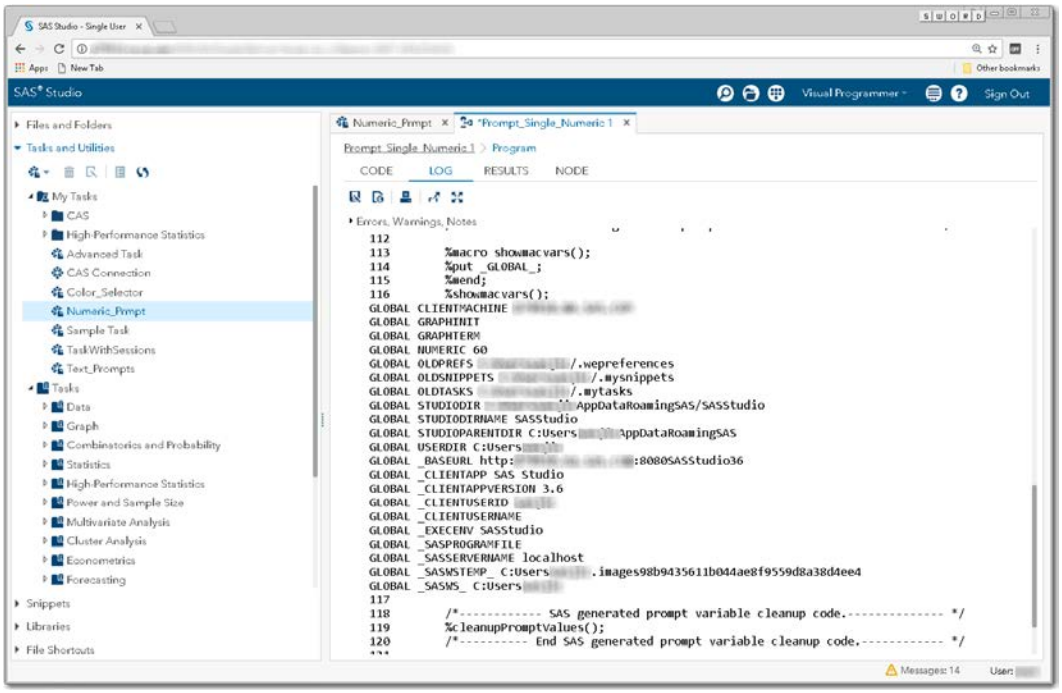
Display 127 - Commented Out %setPromptValues Macro Call

To run your flow with a different numeric value than the default value, open the Prompt_Single_Numeric node and specify a different number.



Display 128 – Running the Numeric Input Task

When you run the process flow, the global Numeric variable is set to the value specified in the task.

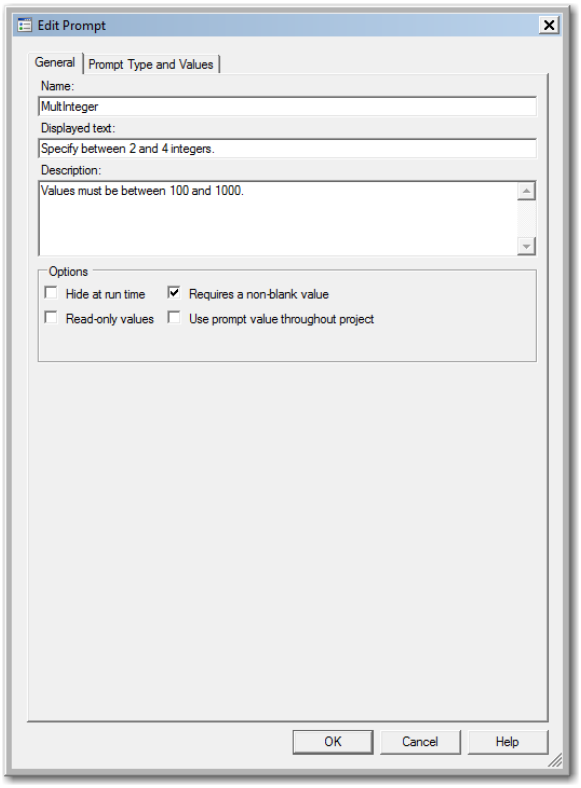


Display 129 - Numeric Prompt Variable with Updated Value

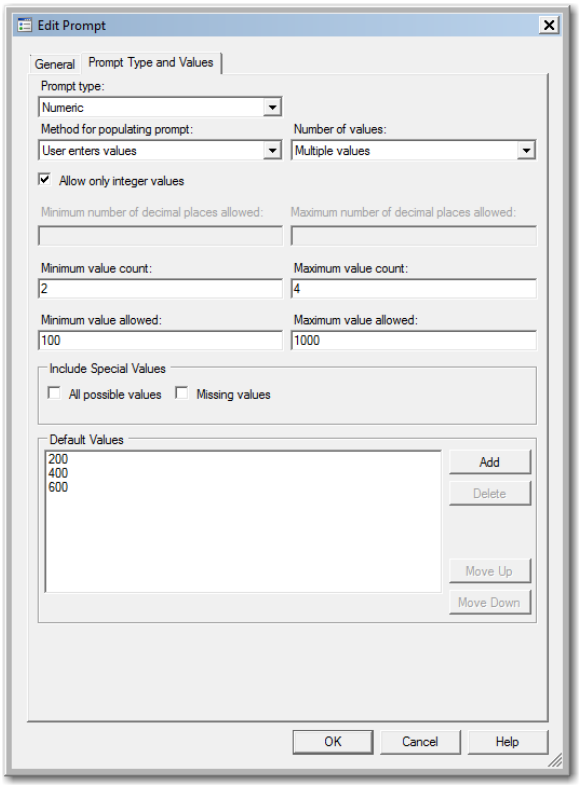
Multiple Numerics

SAS Enterprise Guide

In this example, a multiple values numeric prompt named MultInteger is defined as shown in the following two displays.

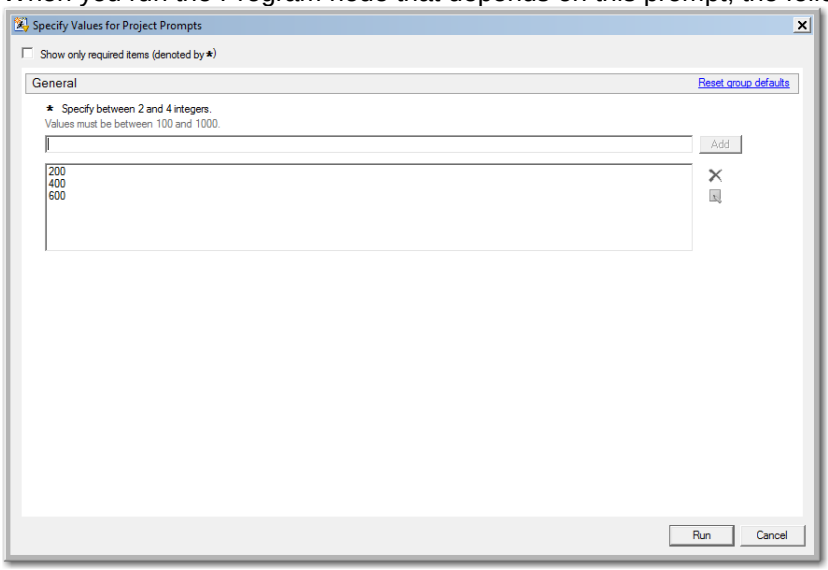


Display 130 - General Properties for Numeric Multiple Values Prompt



Display 131 - Type and Values for Numeric Multiple Values Prompt

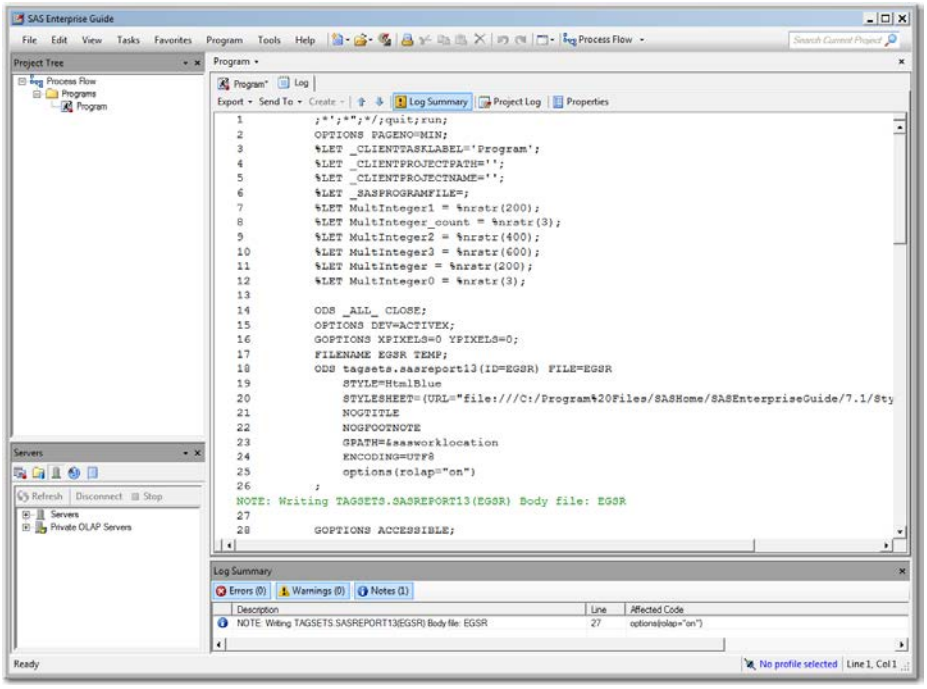
When you run the Program node that depends on this prompt, the following dialog box appears.



Display 132 - Multiple Numeric Values Prompt in Prompt Dialog Box

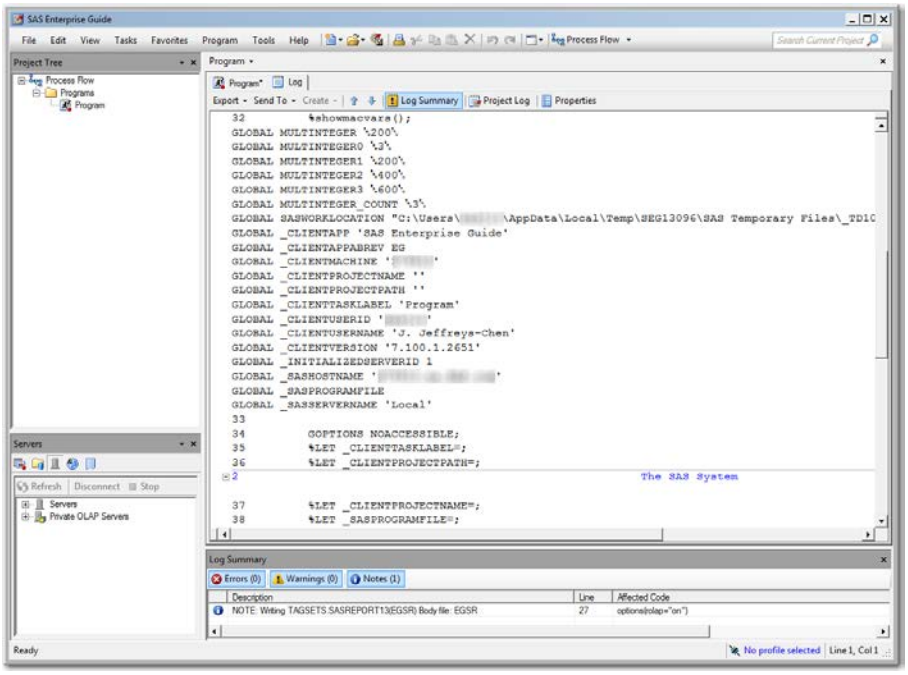
If the user leaves the default value in the multiple numeric value prompt field, the following code is generated by SAS Enterprise Guide in the Program node that depends on the prompt.

The %LET statements assign the values specified in the prompt dialog box to the MultInteger* macro variables.



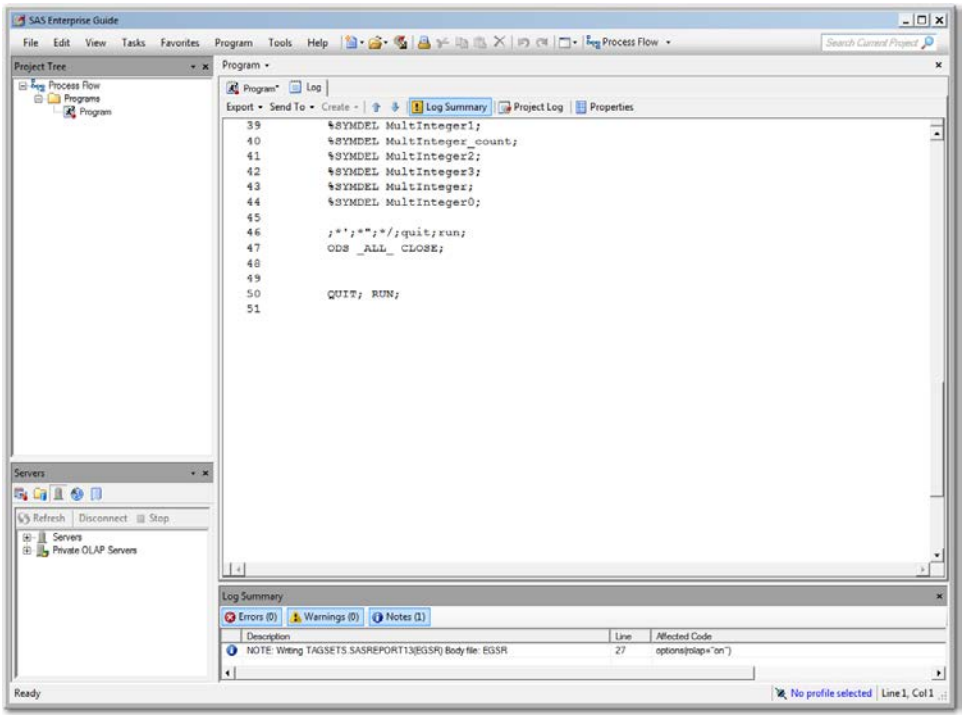
Display 133 - %LET Statements for Multiple Numeric Values Prompt

The log of the Program node using the prompt definition displays the values of the global variables created by the prompt.



Display 134 - Macro Variables for Multiple Numeric Values Prompt

Because the **Use prompt value throughout project** option is not checked for this prompt, the %SYMDEL statements remove the macro variables at the end of the program.



Display 135 - %SYMDEL Statements Remove MultInteger* Macro Variables

SAS Studio

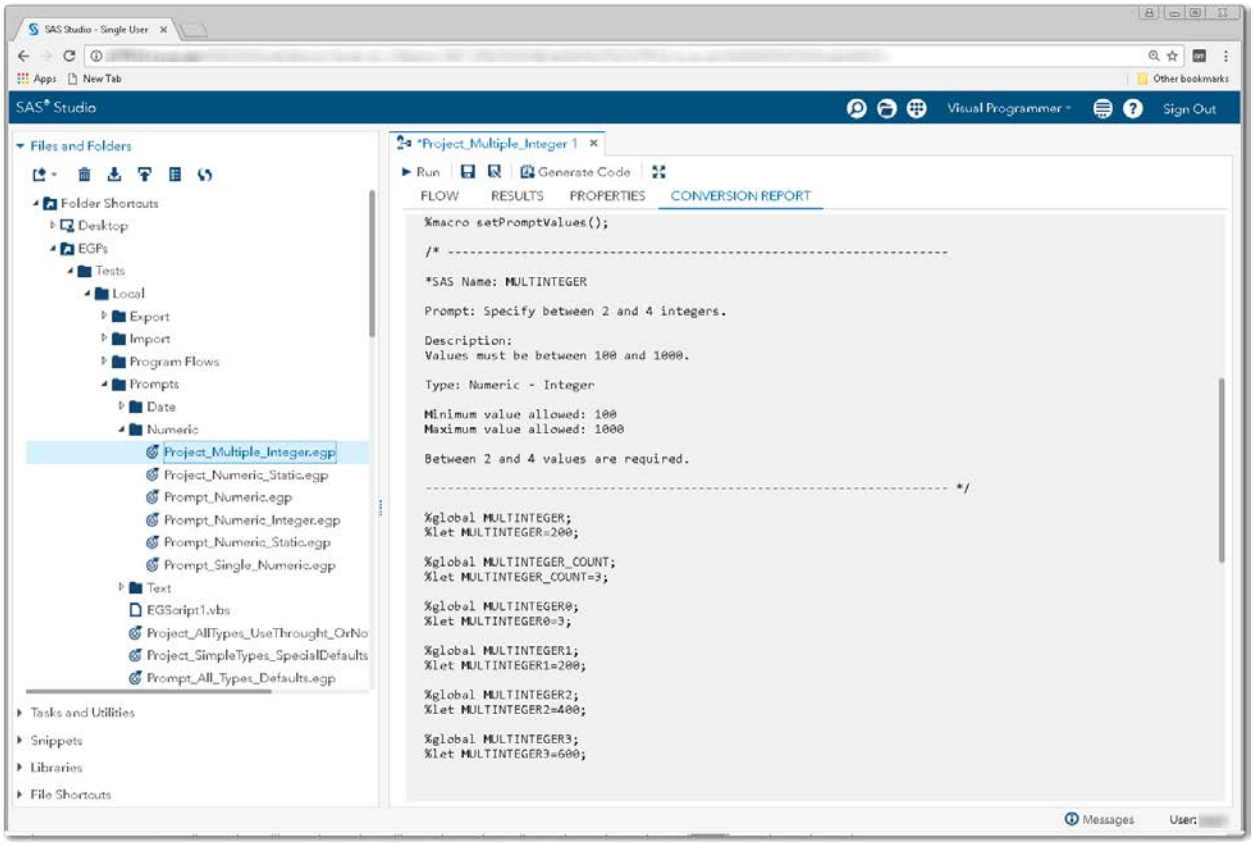
The following display shows the code that is added to the converted Program node for the multiple numeric values prompt in SAS Enterprise Guide.

These global variables are created:

- MULTINTEGER
- MULTINTEGER_COUNT
- MULTINTEGER0
- MULTINTEGER1
- MULTINTEGER2
- MULTINTEGER3

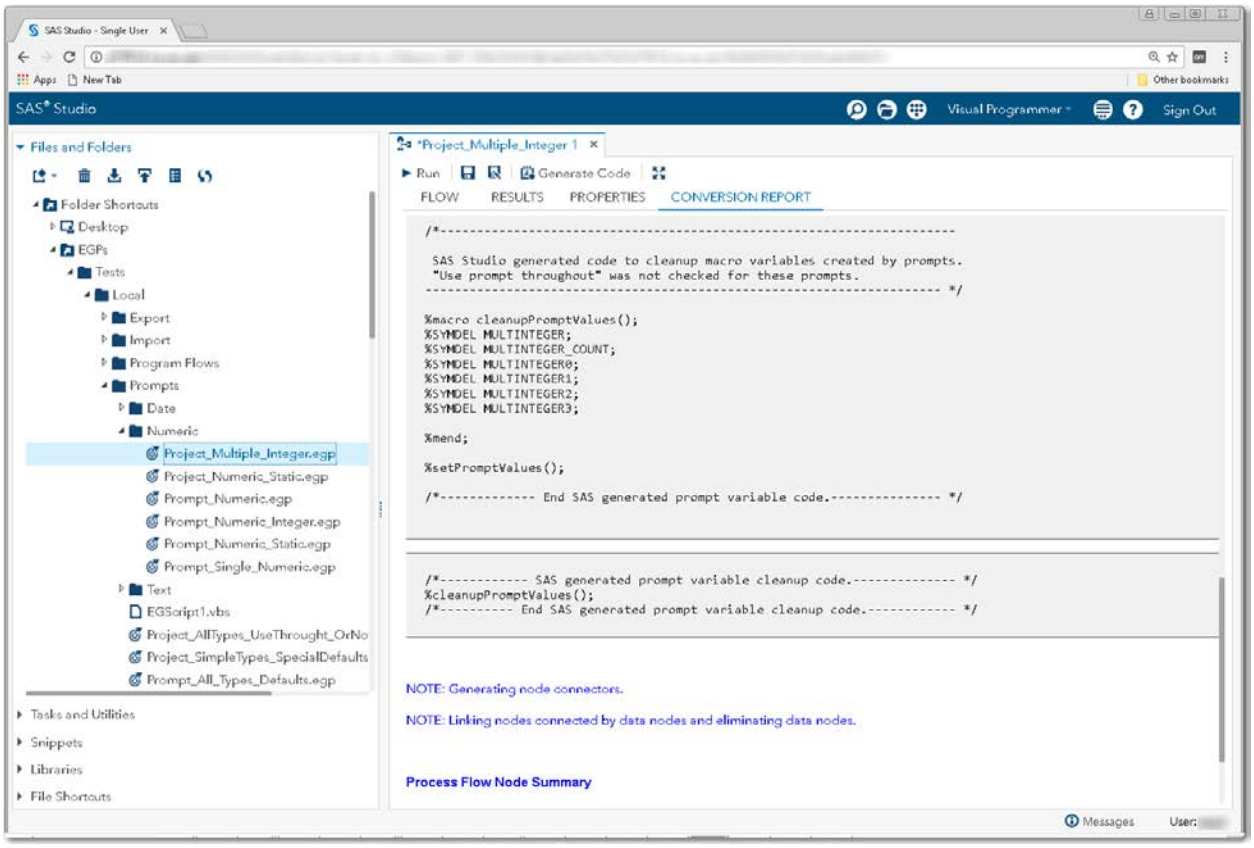
A %LET statement assigns the default values to the MULTINTEGER1, MULTINTEGER2, and MULTINTEGER3 variables.

If you want to run your process flow using different values for the MULTINTEGER prompt, you must manually update values of the macro variables in the %LET statements. Note that in this example, the MULTINTEGER_COUNT and MULTINTEGER_0 variables must reflect the number of text selections you want your program to process, and the MULTINTEGERn variables must be in sequential order.



Display 136 - Code for Multiple Numeric Values Prompt

Because the **Use prompt value throughout project** option is not checked, the %SYMDEL statements remove the MULTINTEGER macro variables.

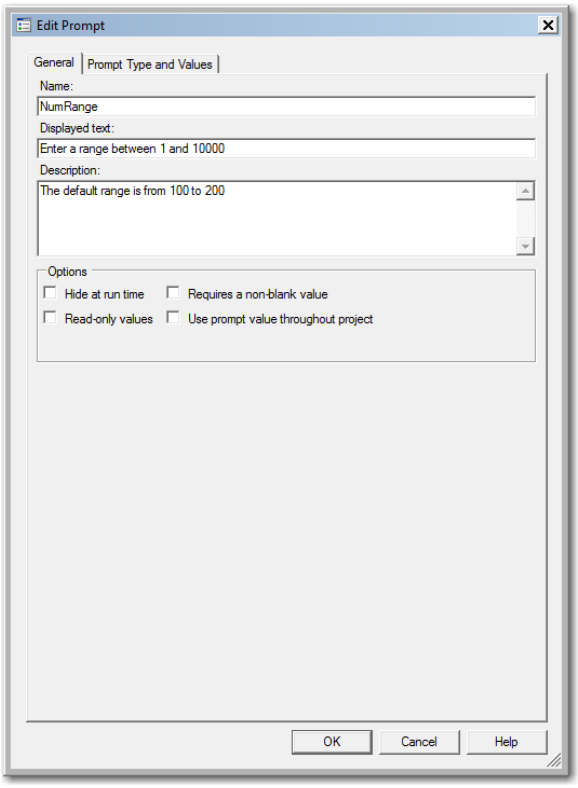


Display 137 - %SYMDEL Statements Remove the MULTINTEGER* Macro Variables

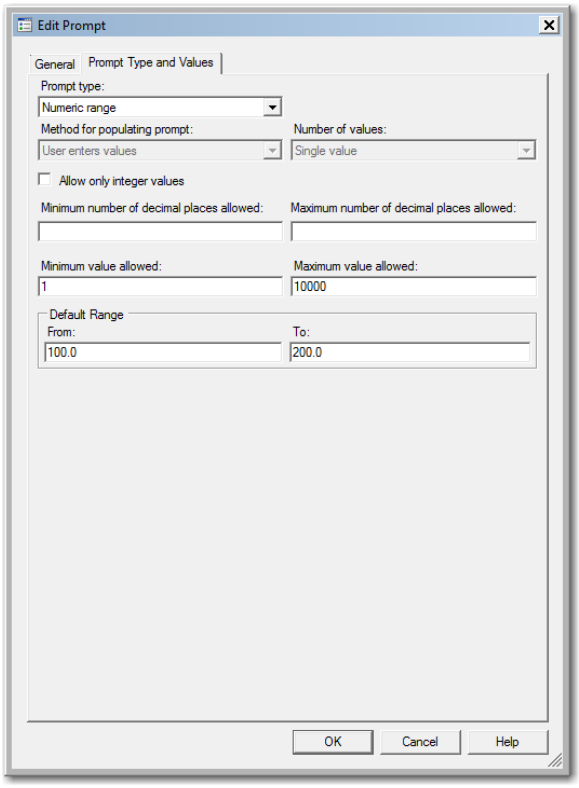
Numeric Range

SAS Enterprise Guide

In this example, a numeric range prompt named NumRange is defined as shown in the following two displays.

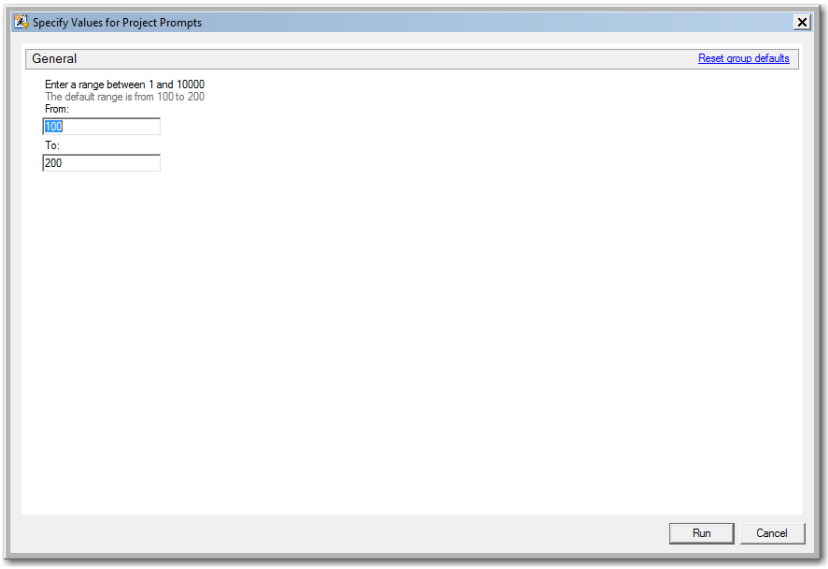


Display 138 - General Properties for Numeric Range Prompt



Display 139 - Type and Values for Numeric Range Prompt

When you run the Program node that depends on this prompt, the following dialog box appears.

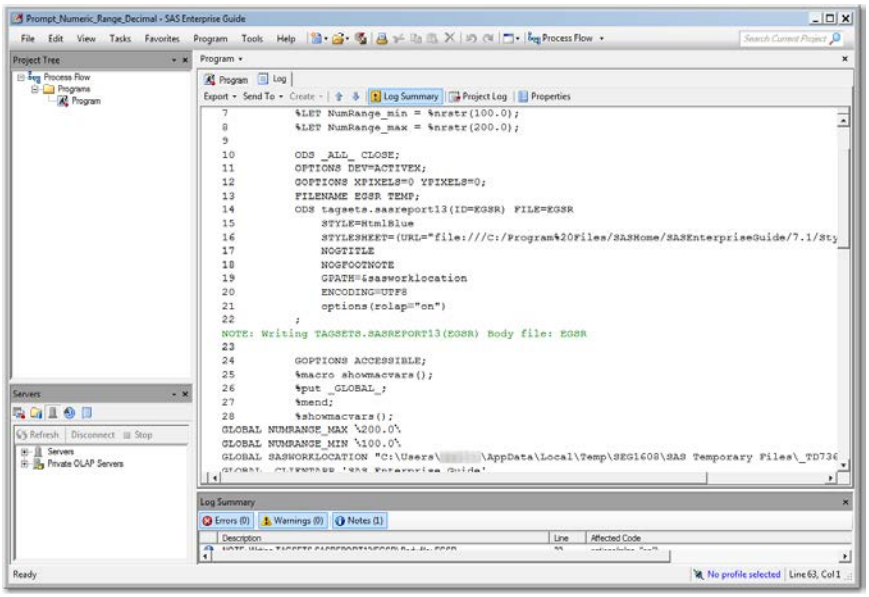


Display 140 - Numeric Range Prompt in Prompt Dialog Box

If the user leaves the default values in the numeric range value prompt fields, the following code is generated by SAS Enterprise Guide in the Program node that depends on the prompt.

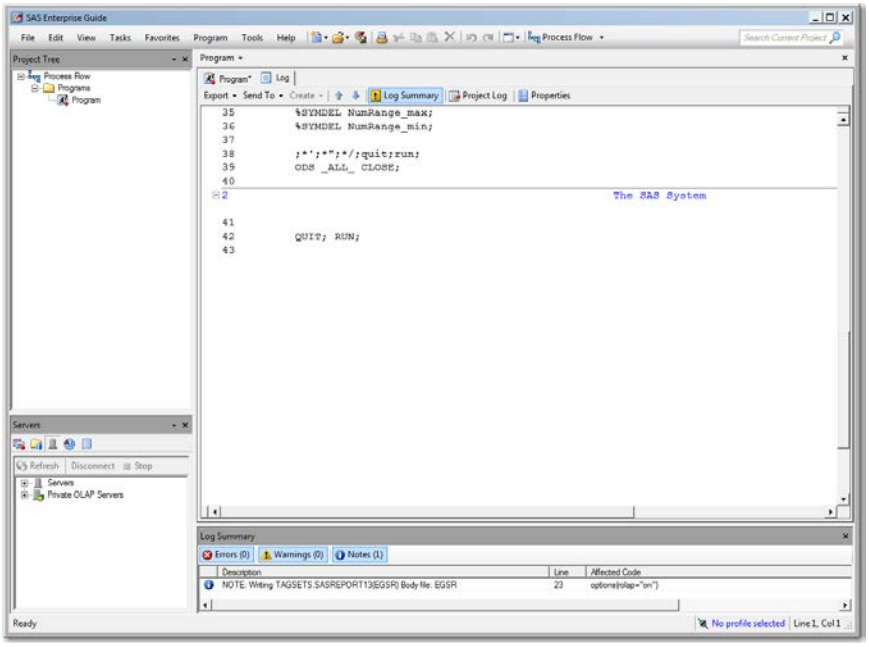
The %LET statements assign the values specified in the prompt dialog box to the NumRange* macro variables.

The log of the Program node using the prompt definition displays the values of the global variables created by the prompt.



Display 141 - Global Macro Variables and %LET Statements for Numeric Range Prompt

Because the **Use prompt value throughout project** option is not checked for this prompt, the %SYMDEL statements remove the macro variables at the end of the program.

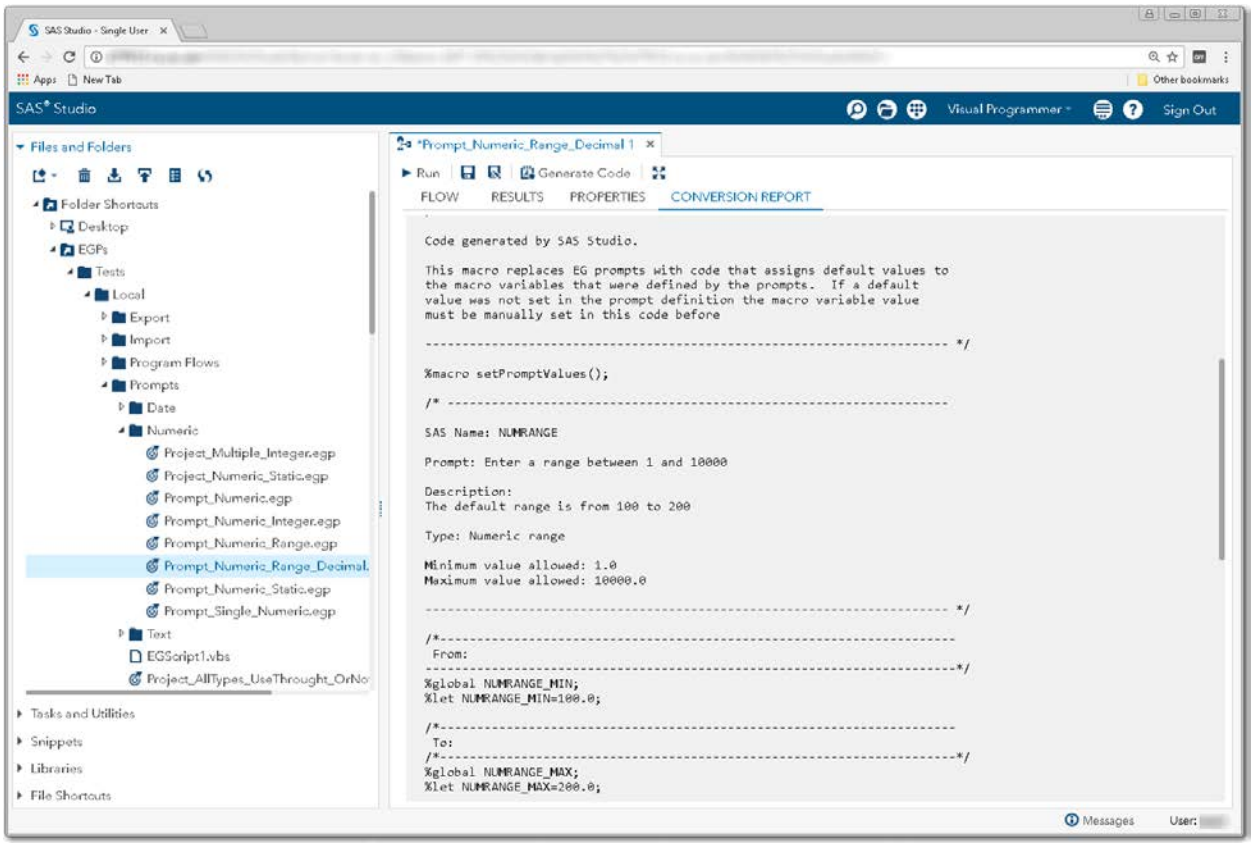


Display 142 - %SYMDEL Statements Remove NumRange* Macro Variables

SAS Studio

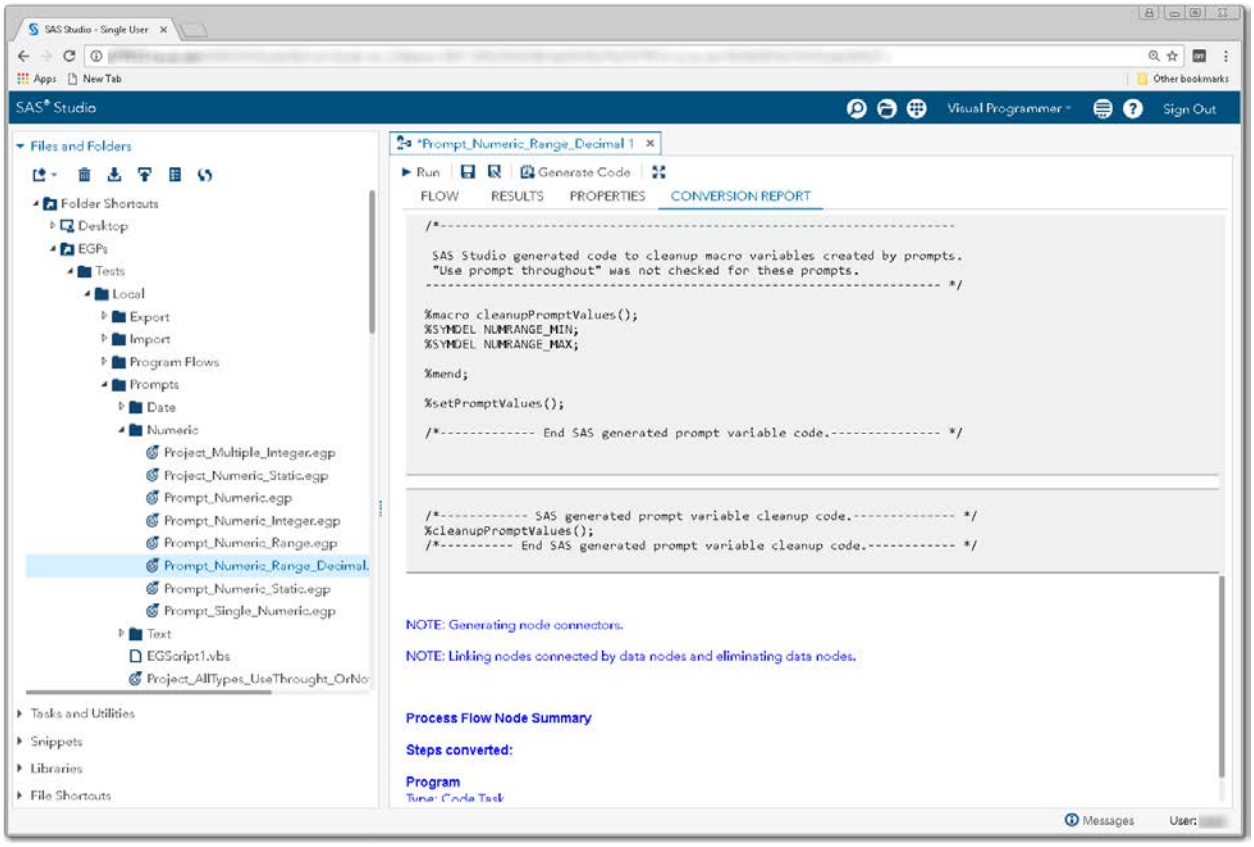
The following display shows code that is added to the converted Program node for the numeric range prompt in SAS Enterprise Guide.

Global variables named NUMRANGE_MIN and NUMRANGE_MAX are created. The %LET statements assign the default values to NUMRANGE_MIN and NUMRANGE_MAX. If you want to run your process flow using different values for the NUMRANGE prompt, you must manually update the values in the macro variables in the %LET statement.



Display 143 – Code for Numeric Range Prompt

Because the **Use prompt value throughout project** option is not checked, the %SYMDEL statements remove the NUMRANGE* macro variables.

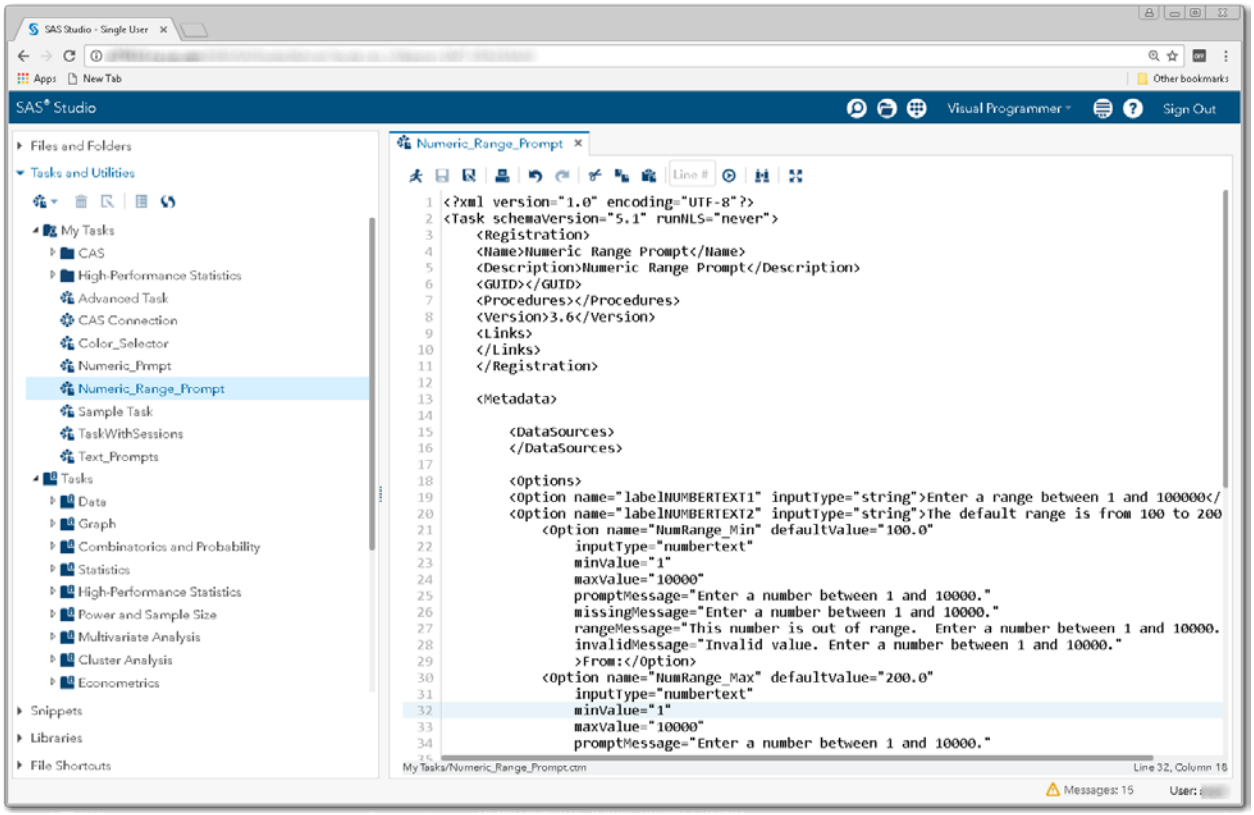


Display 144 - %SYMDEL Statements Remove NUMRANGE* Macro Varaible

Note: There is a SAS Studio bug that causes numeric range prompt information to not be processed when the **Allow only integer values** option is checked.

Substituting a SAS Studio Task a for Numeric Range Prompt

1. Create a SAS Studio task with a control that represents the prompt for a NumRange.
   - Add labels and two numbertext input controls. For the names of the input controls, use the prompt names of NumRange_Min and NumRange_Max.
   - Set the default values to the default values shown in the generated `setPromptValues()` macro in the converted Program node.
   - Change the strings of the input controls to match the strings specified in the prompt.

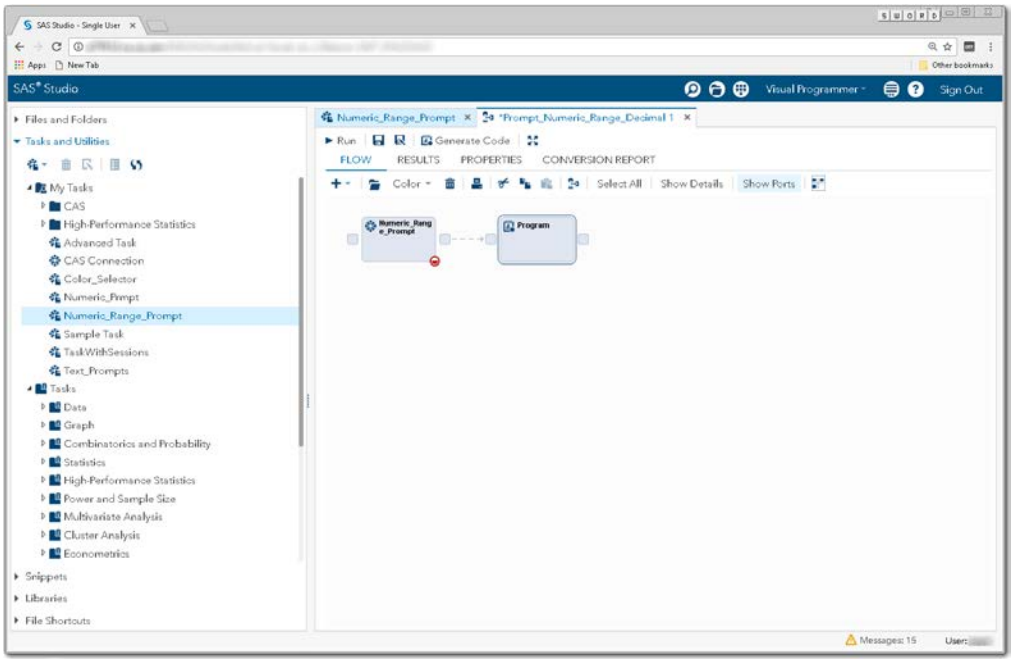


Display 145 - Replacement Task for Numeric Range Prompt

The following code is an example of a task that could be used to replace the numeric range prompt.

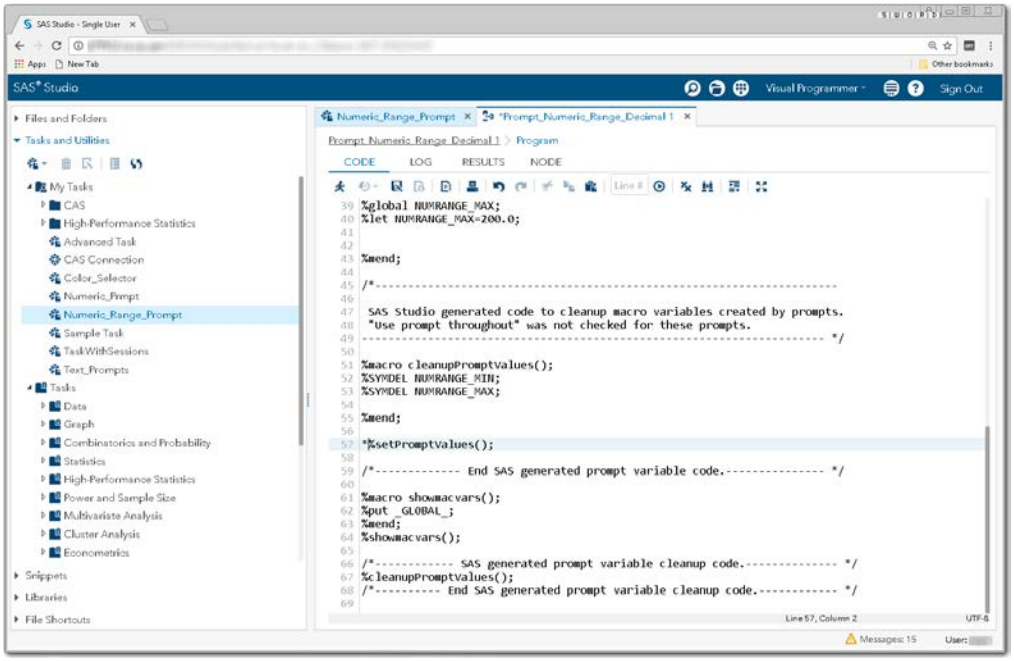
```
<?xml version="1.0" encoding="UTF-8"?>
<Task schemaVersion="5.1" runNLS="never">
       <Registration>
       <Name>Numeric Range Prompt</Name>
       <Description>Numeric Range Prompt</Description>
       <GUID></GUID>
       <Procedures></Procedures>
       <Version>3.6</Version>
       <Links></Links>
       </Registration>
       <Metadata>
              <DataSources>
              </DataSources>
              <Options>
                     <Option name="labelNUMBERTEXT1" inputType="string">
                            Enter a range between 1 and 100000
                     </Option>
                     <Option name="labelNUMBERTEXT2" inputType="string">
                            The default range is from 100 to 200
                     </Option>
                     <Option name="NumRange_Min" defaultValue="100.0"
                            inputType="numbertext"
                            minValue="1"
                            maxValue="10000"
                            promptMessage="Enter a number between 1 and 10000."
                            missingMessage="Enter a number between 1 and 10000."
                            rangeMessage=
                    "This number is out of range.  Enter a number between 1 and 10000."
                            invalidMessage=
                               "Invalid value. Enter a number between 1 and 10000.">
                         From:
                     </Option>
                     <Option name="NumRange_Max" defaultValue="200.0"
                            inputType="numbertext"
                            minValue="1"
                            maxValue="10000"
                            promptMessage="Enter a number between 1 and 10000."
                            missingMessage="Enter a number between 1 and 10000."
                            rangeMessage=
                    "This number is out of range.  Enter a number between 1 and 10000."
                            invalidMessage=
                              "Invalid value. Enter a number between 1 and 10000.">
                          To:
                     </Option>
              </Options>
       </Metadata>
       <UI>
              <OptionItem option="labelNUMBERTEXT1"/>
              <OptionItem option="labelNUMBERTEXT2"/>
              <OptionItem option="NumRange_Min"/>
              <OptionItem option="NumRange_Max"/>
       </UI>
       <CodeTemplate>
              <![CDATA[
%global NumRange_Min;
%let NumRange_Min=$NumRange_Min;
%global NumRange_Max;
%let NumRange_Max=$NumRange_Max;
              ]]>
       </CodeTemplate>
</Task>
```

2. Save the prompt replacement task to your **My Tasks** folder.
3. Drag the task from **My Tasks** into your converted process flow.
4. Link the output port of the task to the input port of the converted Program node.



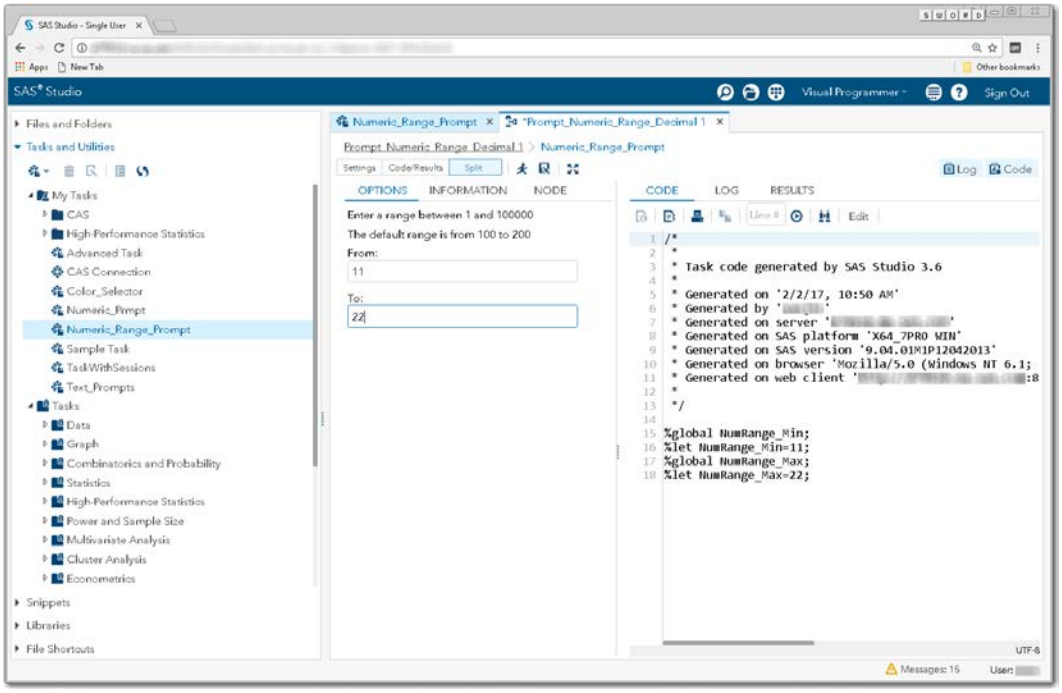
Display 146 – Numeric Range Input Task Linked to Program Node

5. Comment out the `%setPromptValues()` macro call from the converted Program node. The macro code generated by the numeric range input task replaces this code.



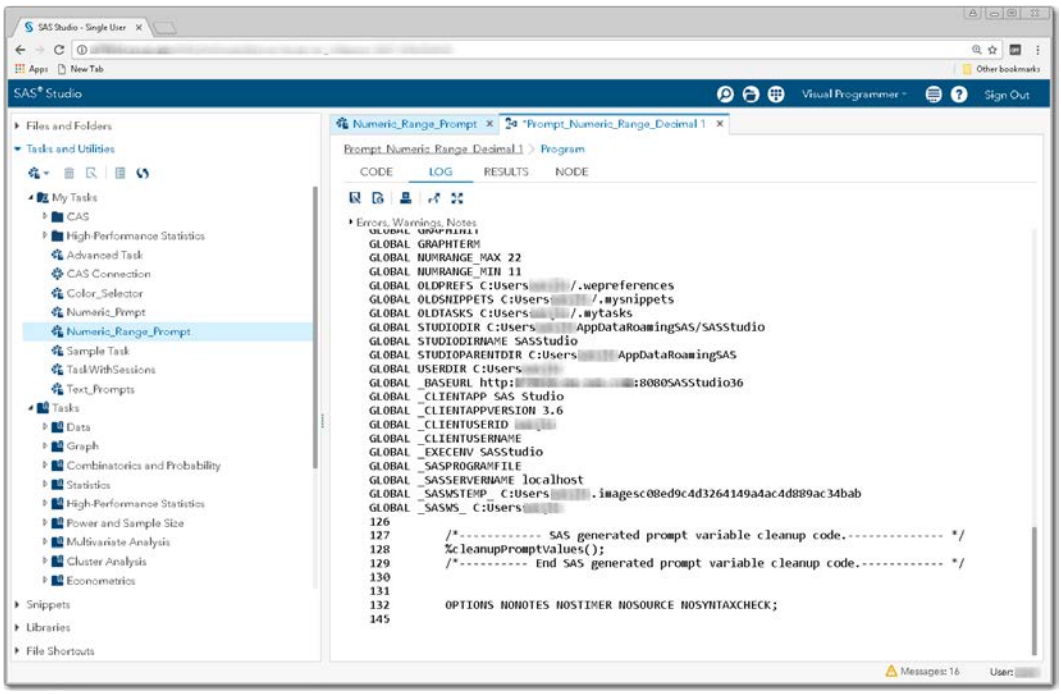
Display 147 - Commented Out %setPromptValues Macro Call

To run your flow with a different numeric value than the default value, open the Numeric_Range_Prompt node and specify a different numeric range.



Display 148 - Running the Numeric Range Input Task

When you run the process flow, the global NumRange* variables are set to the values specified in the task.

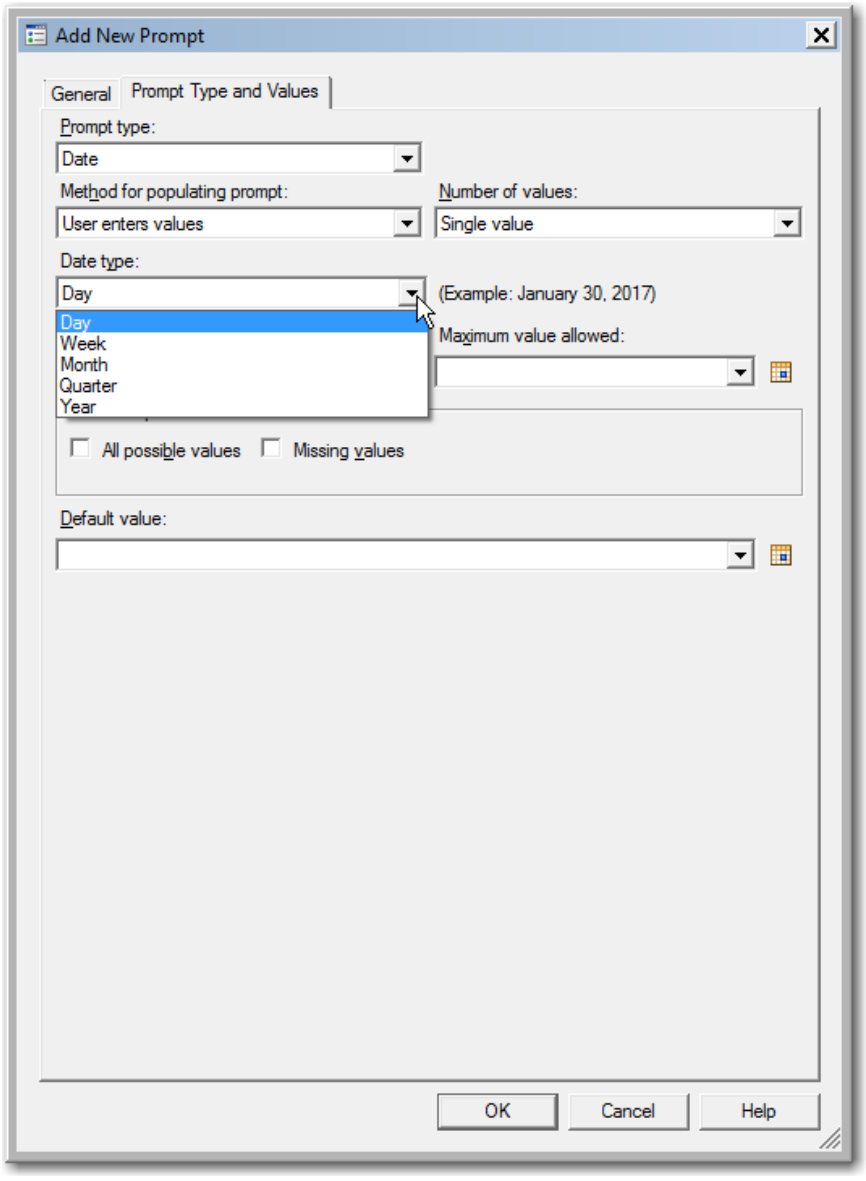


Display 149 – Numeric Range Prompt Variables with Updated Values

Date

For Date prompts, SAS Enteprise Guide supports many different types of dates including:

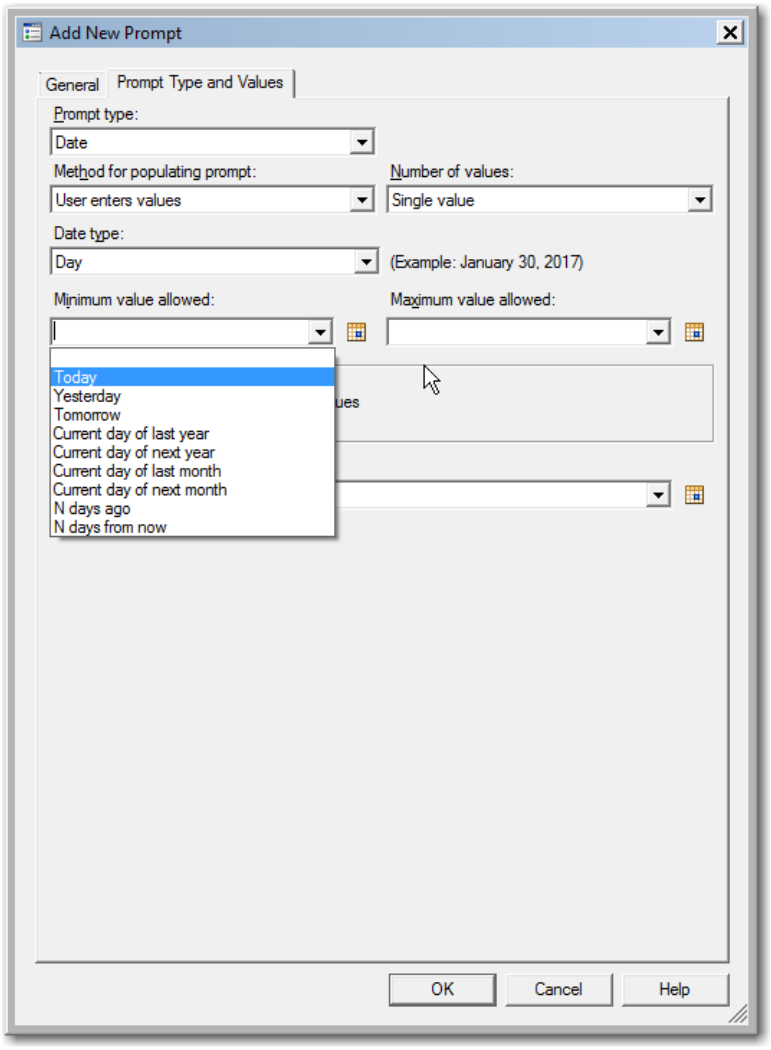
- Day
- Week
- Month
- Quarter
- Year



Display 150 - Types of Date Prompts

Explanation of Date Macro Variable

For each of the date types, you can set defaults, minimums, and maximums in the definitions. In the prompt dialog box, you can also specify absolute values or common values such as “Today”, “Tomorrow”, and so on.



Display 151 – Common Values for Dates

Macro variables generated by SAS Enterprise Guide for date prompts include the basic prompt name, name_rel, and name_label. If the prompt is a range, there is also a name_end macro variable.

- The name macro variable will contain the absolute date specified or the date of the common value, such as 30Jan2017

  If the prompt is a period type prompt (such as Quarter), this value will be the beginning of the period. For example, “First Quarter” would have a date of 01Jan2017.

- The name_rel macro variable will contain a code for the common value. For example, “D0D” represents “Today”.

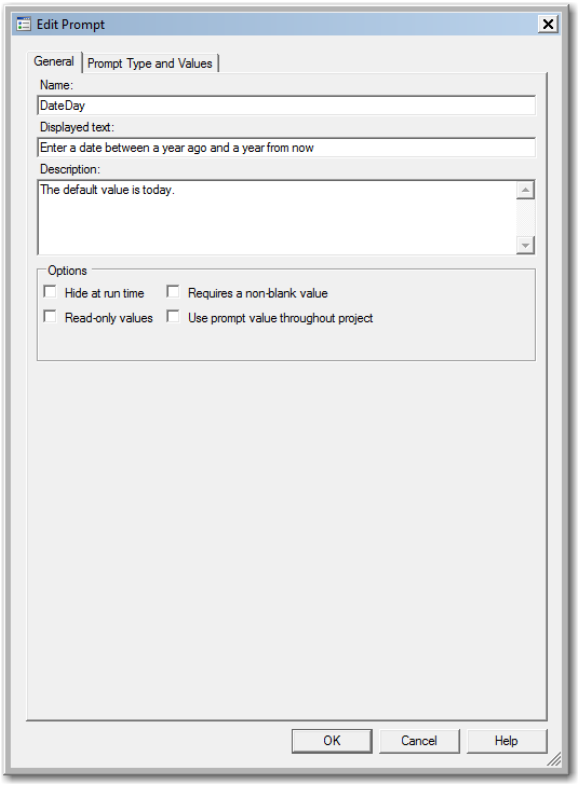
- The name_label macro variable will contain the common value such as “Today” or “Tomorrow”.

- The name_end macro variable will contain the date for the end of the period. For example, for “First Quarter” the end macro variable would have a date of 30Mar2017.

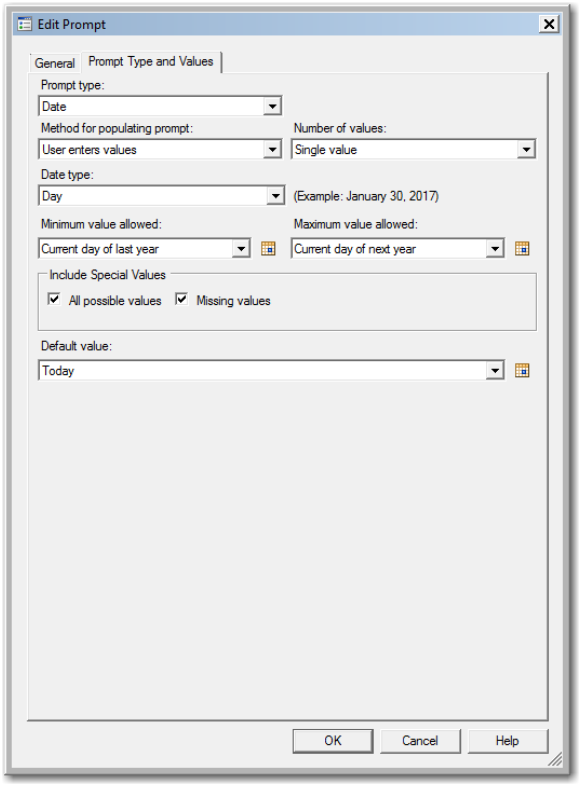
Single Date

SAS Enterprise Guide

In this example, a single date prompt named DateDay is defined as shown in the following two displays.

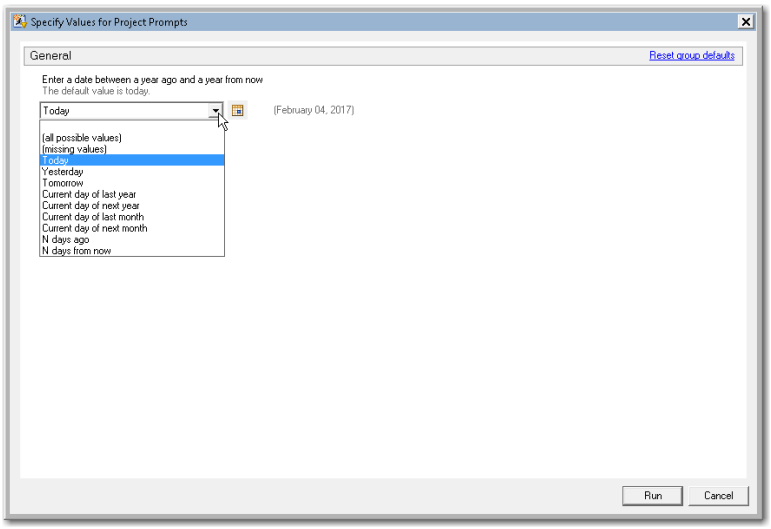


Display 152 - General Properties for Date Prompt



Display 153 - Type and Values for Date Prompt

When you run the Program node that depends on the prompt, the following prompt dialog box appears.

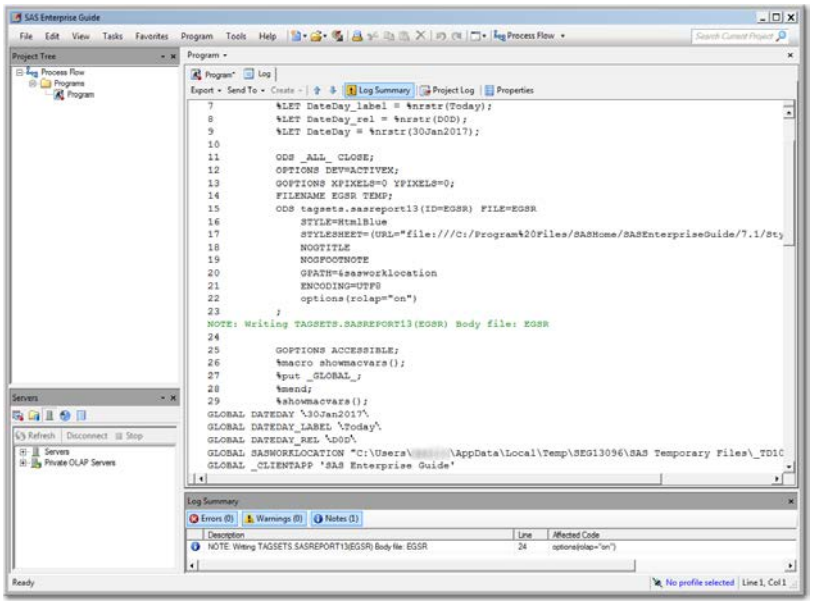


Display 154 - Date Prompt in Prompt Dialog Box

If the user leaves the default value in the date prompt field, the following code is generated by SAS Enterprise Guide in the Program node that depends on the prompt.

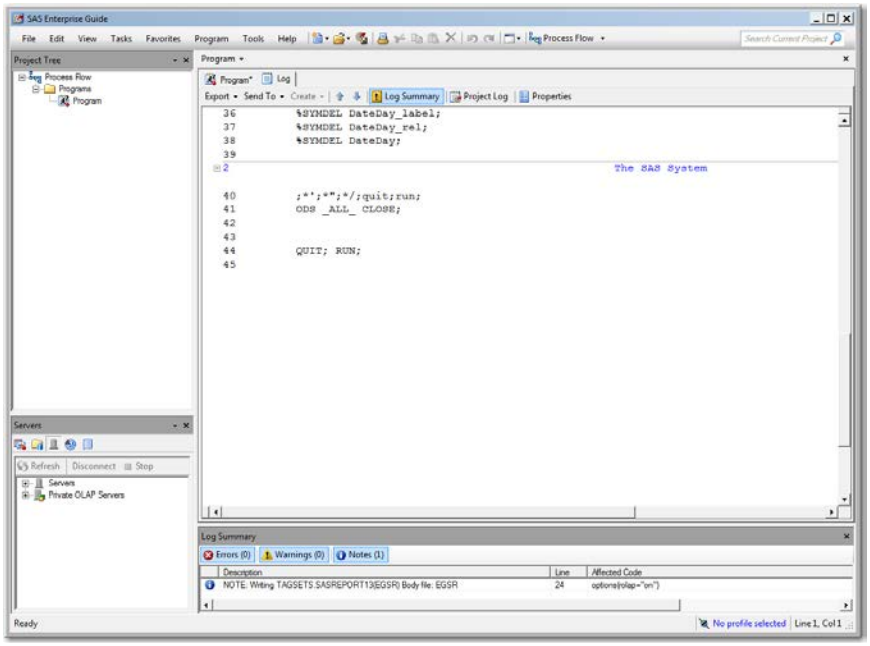
The %LET statements assign the value specified in the prompt dialog box to the DateDay* macro variables.

The log of the Program node using the prompt definition displays the value of the global variable created by the prompt.



Display 155 - %LET Statements for Single Date Prompt

Because the **Use prompt value throughout project** option is not checked for this prompt, the %SYMDEL statements remove the macro variables at the end of the program.



Display 156 - %SYMDEL Statements Remove DateDay* Macro Variables

SAS Studio

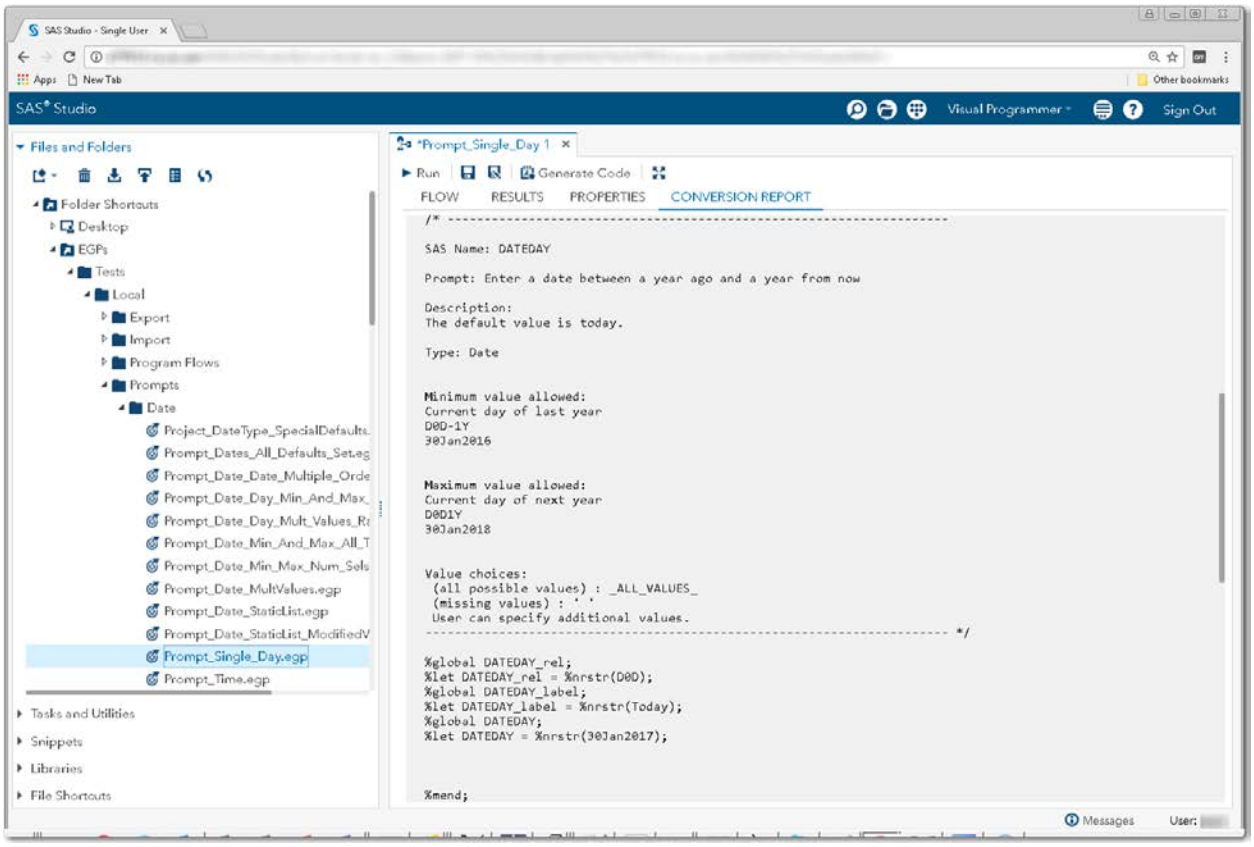
The following display shows the code that is added to the converted Program node for the date prompt in SAS Enterprise Guide.

These global date macro variables are created:

- DATEDAY_rel
- DATEDAY_label
- DATEDAY

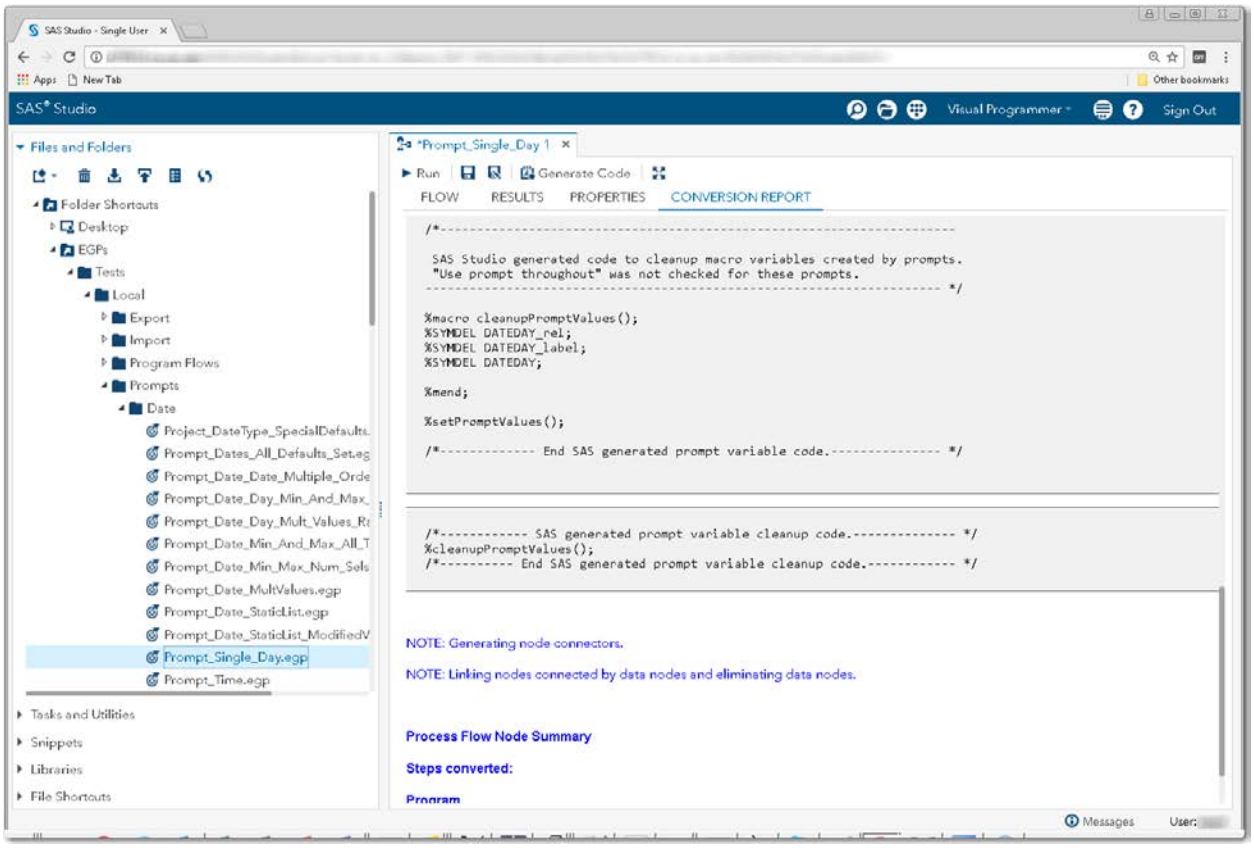
The %LET statements assign the default value to the DATEDAY* macro variables.

If you want to run your process flow using different values for the DATEDAY prompt, you must manually update the values of the macro variables in the %LET statements.



Display 157 - Macro Code for Date Prompt

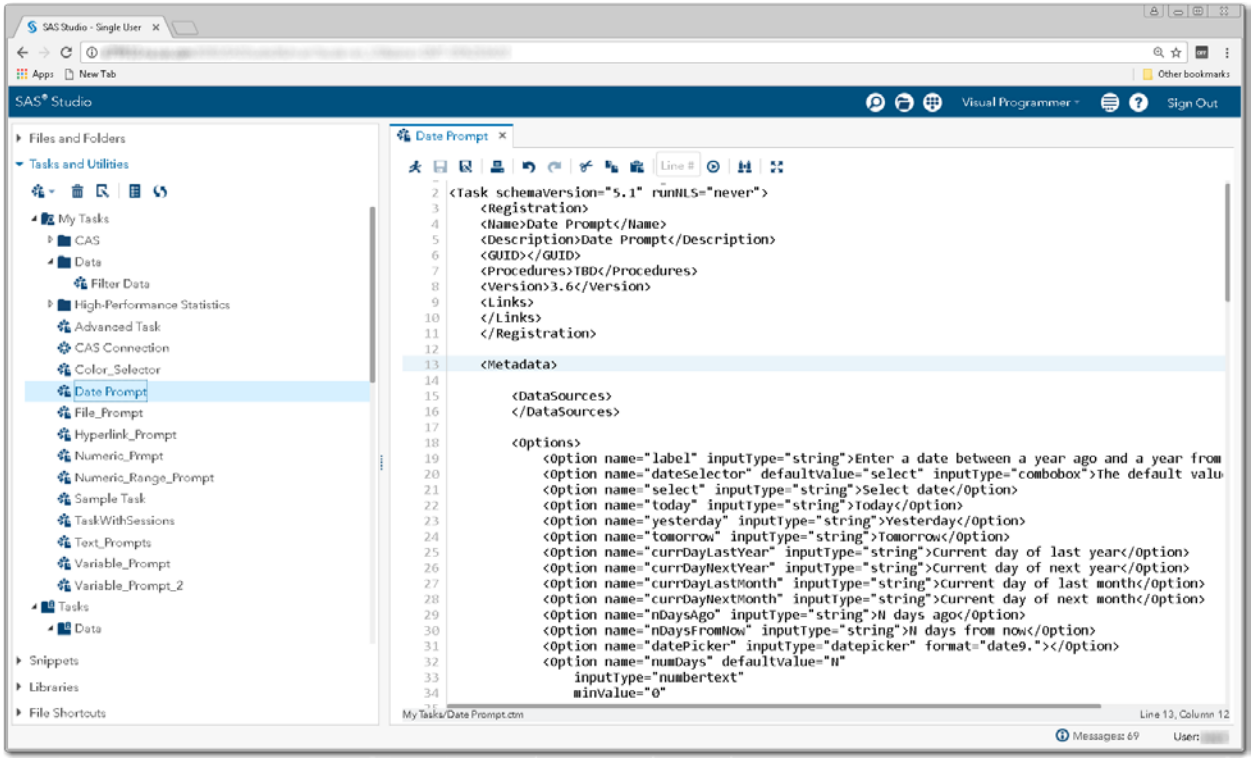
Because the **Use prompt value throughout project** option is not checked, the %SYMDEL statements remove the DATEDAY macro variables.



Display 158 - %SYMDEL Statements Remove the DATEDAY* Macro Variables

Substituting a SAS Studio Task for a Date Range Prompt

1. Create a SAS Studio task with a control that represents the Day type prompt for a Date.
   - Add controls as shown in the Date Prompt task.
   - Set the default value to the default value shown in the generated `setPromptValues()` macro in the converted Program node.
   - Change the strings for the input controls to match the strings specified in the prompt.



Display 159 - Replacement Task for Date Prompt

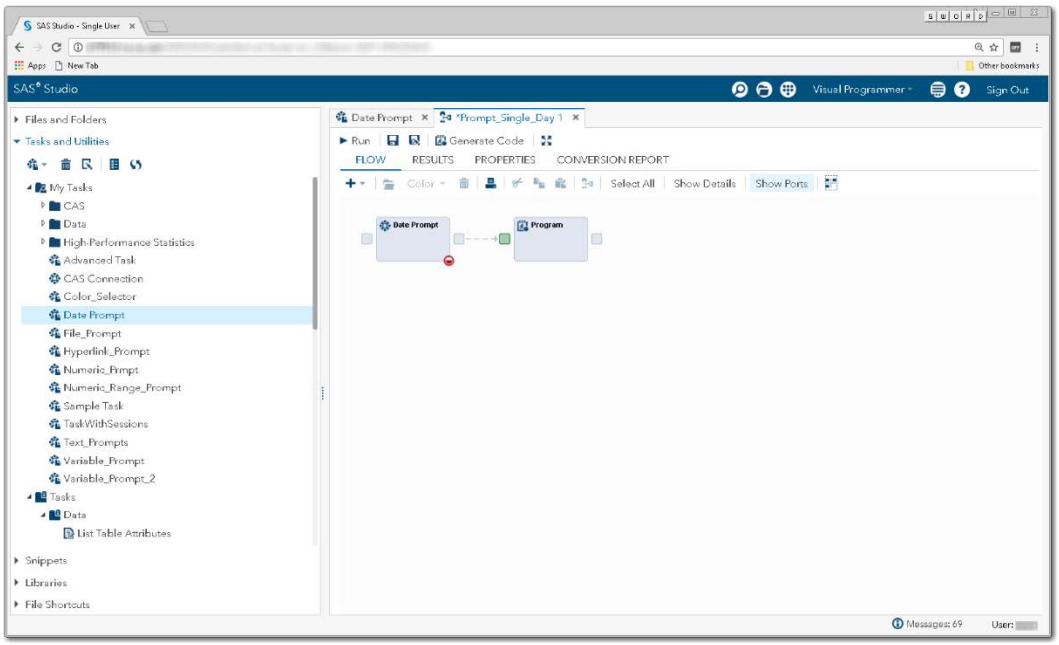
The following code is an example of a task that could be used as the date day prompt.

```
<?xml version="1.0" encoding="UTF-8"?>
<Task schemaVersion="5.1" runNLS="never">
       <Registration>
       <Name>Date Prompt</Name>
       <Description>Date Prompt</Description>
       <GUID></GUID>
       <Procedures>TBD</Procedures>
       <Version>3.6</Version>
       <Links></Links>
       </Registration>

       <Metadata>

              <DataSources></DataSources>

              <Options>
                     <Option name="label" inputType="string">
                         Enter a date between a year ago and a year from now
                     </Option>
                     <Option name="dateSelector"
                             defaultValue="select"
                             inputType="combobox">
                         The default value is today
                     </Option>
                     <Option name="select" inputType="string">Select date</Option>
                     <Option name="today" inputType="string">Today</Option>
                     <Option name="yesterday" inputType="string">Yesterday</Option>
                     <Option name="tomorrow" inputType="string">Tomorrow</Option>
                     <Option name="currDayLastYear" inputType="string">
                         Current day of last year
                     </Option>
                     <Option name="currDayNextYear" inputType="string">
                         Current day of next year
                     </Option>
                     <Option name="currDayLastMonth" inputType="string">
                         Current day of last month
                     </Option>
                     <Option name="currDayNextMonth" inputType="string">
                         Current day of next month
                     </Option>
                     <Option name="nDaysAgo" inputType="string">
                         N days ago
                     </Option>
                     <Option name="nDaysFromNow" inputType="string">
                         N days from now
                     </Option>
                     <Option name="datePicker" inputType="datepicker" format="date9.">
                     </Option>
                     <Option name="numDays" defaultValue="N"
                            inputType="numbertext"
                            minValue="0"
                            promptMessage="Enter number of days"
                            missingMessage="Enter number of days"
                            rangeMessage=
                                "This number is out of range.  Enter a positive number."
                           invalidMessage="Invalid value. Enter a positive integer.">
                          Number of days:
                     </Option>
              </Options>

       </Metadata>
```

```
<UI>
    <OptionItem option="label"/>
      <OptionChoice option="dateSelector">
            <OptionItem option="select"/>
            <OptionItem option="today"/>
            <OptionItem option="yesterday"/>
            <OptionItem option="tomorrow"/>
            <OptionItem option="currDayLastYear"/>
            <OptionItem option="currDayNextYear"/>
            <OptionItem option="currDayLastMonth"/>
            <OptionItem option="currDayNextMonth"/>
            <OptionItem option="nDaysAgo"/>
            <OptionItem option="nDaysFromNow"/>
      </OptionChoice>

      <OptionItem option="datePicker"/>
      <OptionItem option="numDays"/>
</UI>
 <Dependencies>
     <Dependency condition="($dateSelector == 'select')">
        <Target action="hide" conditionResult="false" option="datePicker"/>
        <Target action="show" conditionResult="true" option="datePicker"/>
     </Dependency>
     <Dependency condition=
        "(($dateSelector == 'nDaysAgo') || ($dateSelector == 'nDaysFromNow'))">
        <Target action="hide" conditionResult="false" option="numDays"/>
        <Target action="show" conditionResult="true" option="numDays"/>
     </Dependency>
 </Dependencies>

<CodeTemplate>
      <![CDATA[


%global dateday;
%global dateday_rel;
%global dateday_label;

#if ($dateSelector == 'select')
%let dateday = $datePicker;
%let dateday_label = %sysfunc(inputn(&dateday, date9.), WORDDATE20.);
%symdel dateday_rel;
#end

#if ($dateSelector == 'today')
%let dateday = %sysfunc(intnx(Day,%sysfunc(date()),0), DATE9.);
%let dateday_label = Today;
%let dateday_rel = D0D;
#end

#if ($dateSelector == 'tomorrow')
%let dateday = %sysfunc(intnx(Day,%sysfunc(date()),+1), DATE9.);
%let dateday_label = Tomorrow;
%let dateday_rel = D1D;
#end

#if ($dateSelector == 'yesterday')
%let dateday = %sysfunc(intnx(Day,%sysfunc(date()),-1), DATE9.);
%let dateday_label = Yesterday;
%let dateday_rel = D-1D;
#end
```

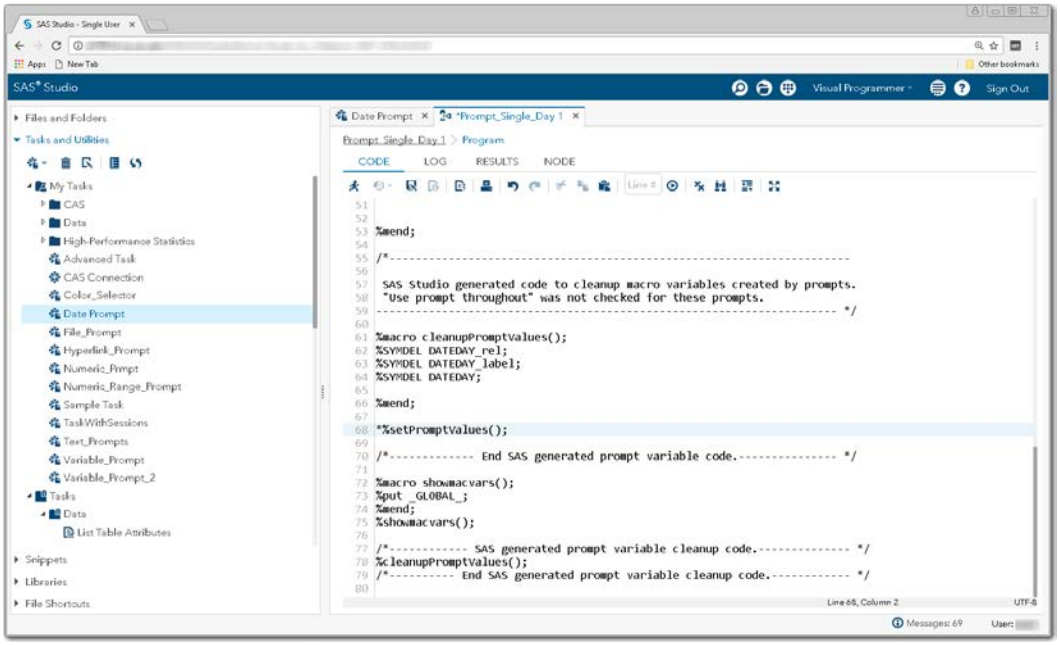
```
#if ($dateSelector == 'currDayLastYear')
%let dateday = %sysfunc(intnx(Year,%sysfunc(date()),-1,s), DATE9.);
%let dateday_label = Current day of last year;
%let dateday_rel = D-1Y;
#end

#if ($dateSelector == 'currDayNextYear')
%let dateday = %sysfunc(intnx(Year,%sysfunc(date()),+1,s), DATE9.);
%let dateday_label = Current day of next year;
%let dateday_rel = D1Y;
#end

#if ($dateSelector == 'currDayLastMonth')
%let dateday = %sysfunc(intnx(Month,%sysfunc(date()),-1,s), DATE9.);
%let dateday_label = Current day of last month;
%let dateday_rel = D-1M;
#end

#if ($dateSelector == 'currDayNextMonth')
%let dateday = %sysfunc(intnx(Month,%sysfunc(date()),+1,s), DATE9.);
%let dateday_label = Current day of next month;
%let dateday_rel = D1M;
#end

#if ($dateSelector == 'nDaysAgo')
%let dateday = %sysfunc(intnx(Day,%sysfunc(date()),-$numDays,s), DATE9.);
%let dateday_label = $numDays days ago;
%let dateday_rel = D-${numDays}D;
#end

#if ($dateSelector == 'nDaysFromNow')
%let dateday = %sysfunc(intnx(Day,%sysfunc(date()),+$numDays,s), DATE9.);
%let dateday_label = $numDays days from now;
%let dateday_rel = D${numDays}D;
#end

            ]]>
      </CodeTemplate>
</Task>
```

2. Save the prompt replacement task to your **My Tasks** folder.
3. Drag the task from **My Tasks** into your converted process flow.
4. Link the output port of the new task to the input port of the converted Program node.



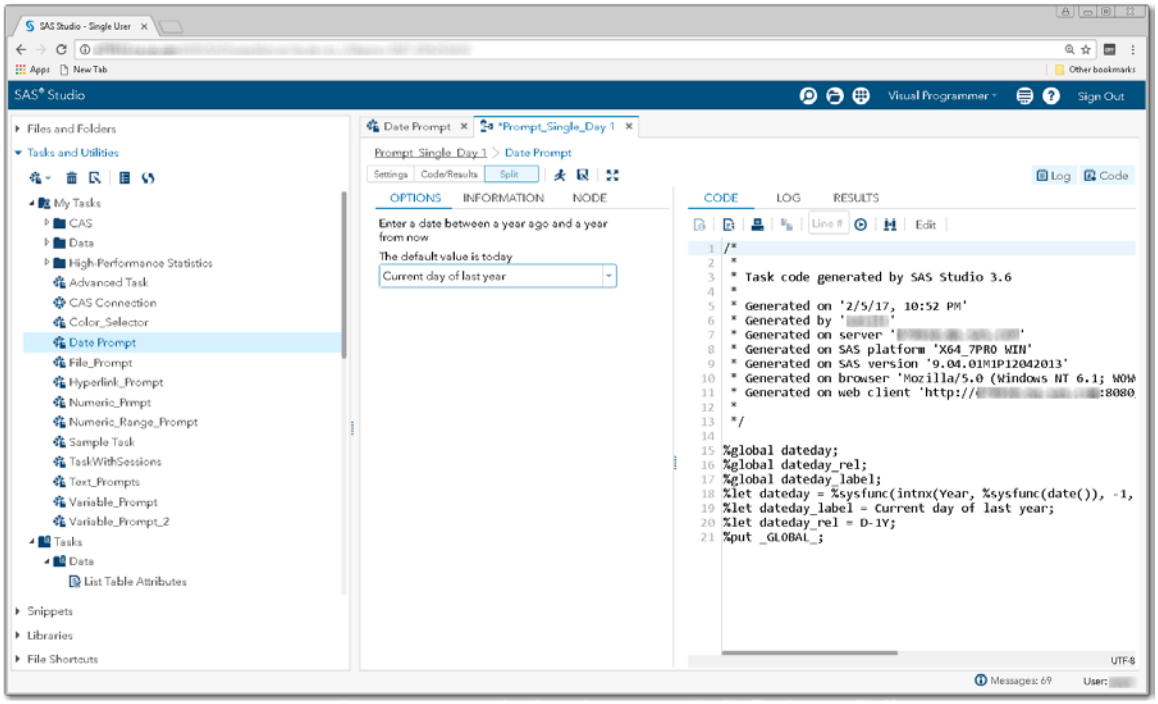
Display 160 – Date Prompt Task Linked to Program Node

5. Comment out the `%setPromptValues()` macro call from the converted Program node. The macro code generated by the Date Prompt input task replaces this code.



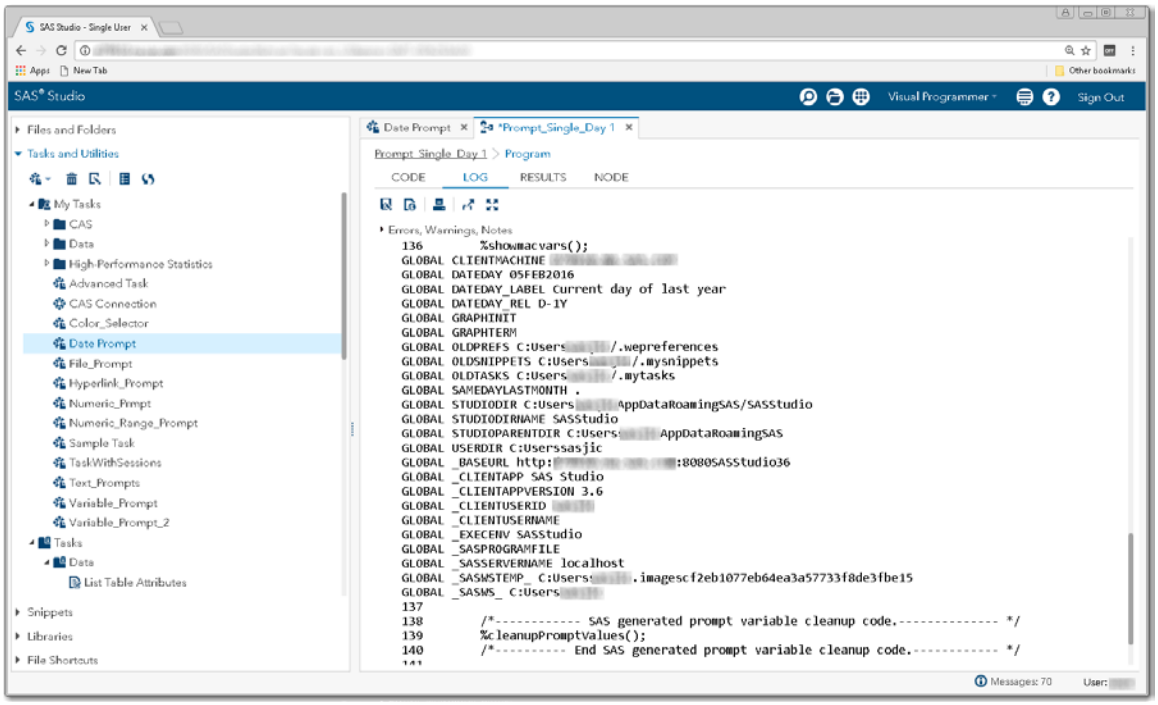
Display 161 – Commented out %setPromptValues in Program Node

To run your flow with a different date value than the default value, open the Date_Prompt node and specify a different date.



Display 162 – Task for Date Prompt

When you run the process flow, the global DateDay* variables are set to the value specified in the Date Prompt task.

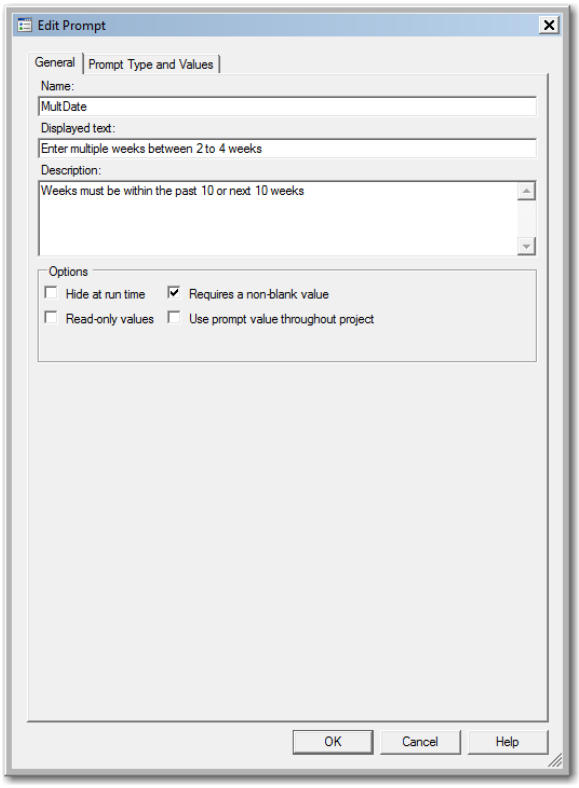


Display 163 – Updated Values for Variables in Date Prompt Task

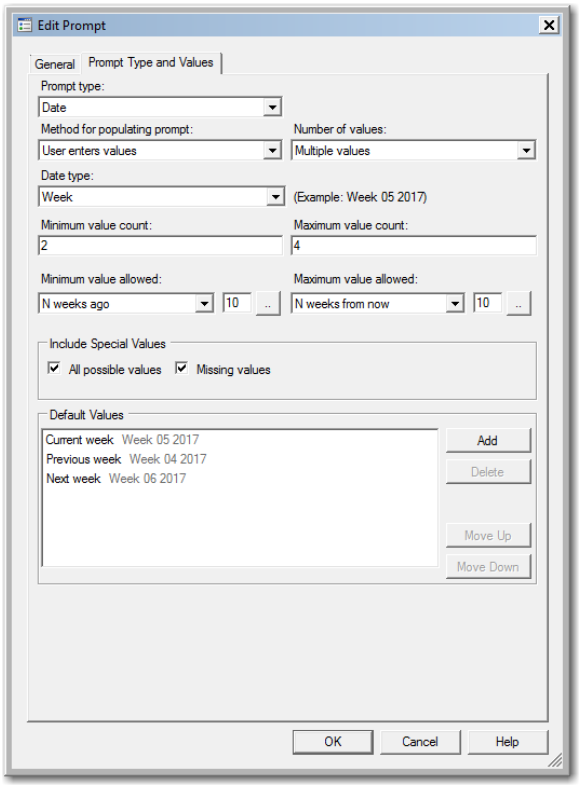
Multiple Dates

SAS Enterprise Guide

In this example, a multiple value date prompt named MultDate is defined as shown in the following two displays.

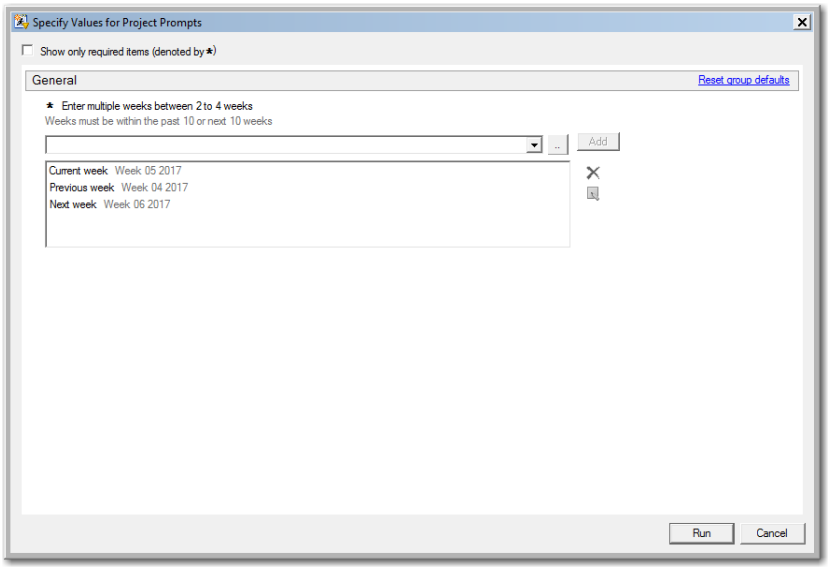


Display 164 - General Properties for Multiple Values Date Prompt



Display 165 - Type and Values for Multiple Values Date Prompt

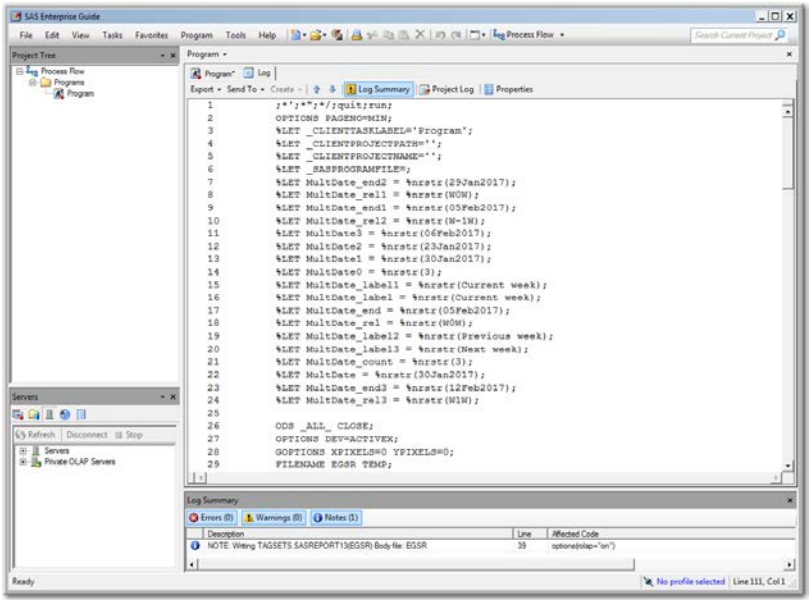
When you run the Program node that depends on the prompt, the following dialog box appears:



Display 166 - Multiple Value Date Prompt in Prompt Dialog Box

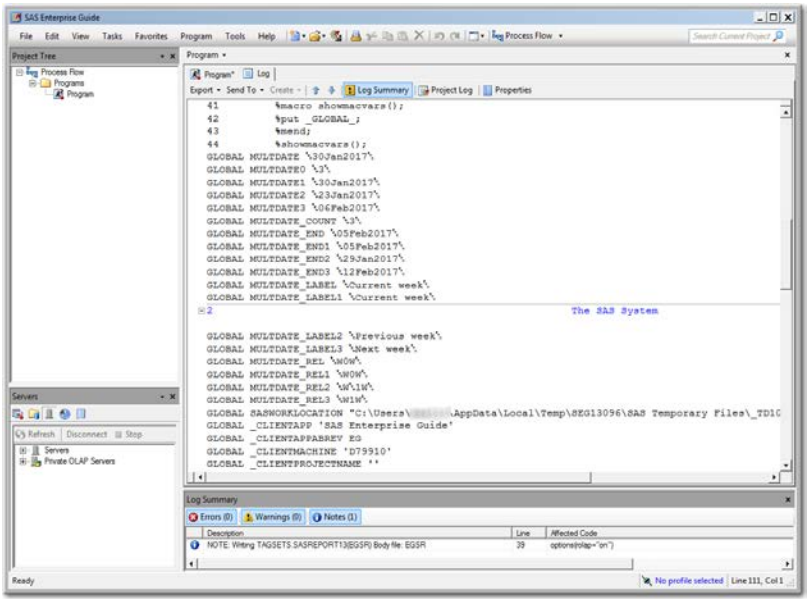
If the user leaves the default value in the multiple date prompt field, the following code is generated by SAS Enterprise Guide in the Program node that depends on the prompt.

The %LET statements assign the values specified in the prompt dialog box to the MultDate* macro variables.



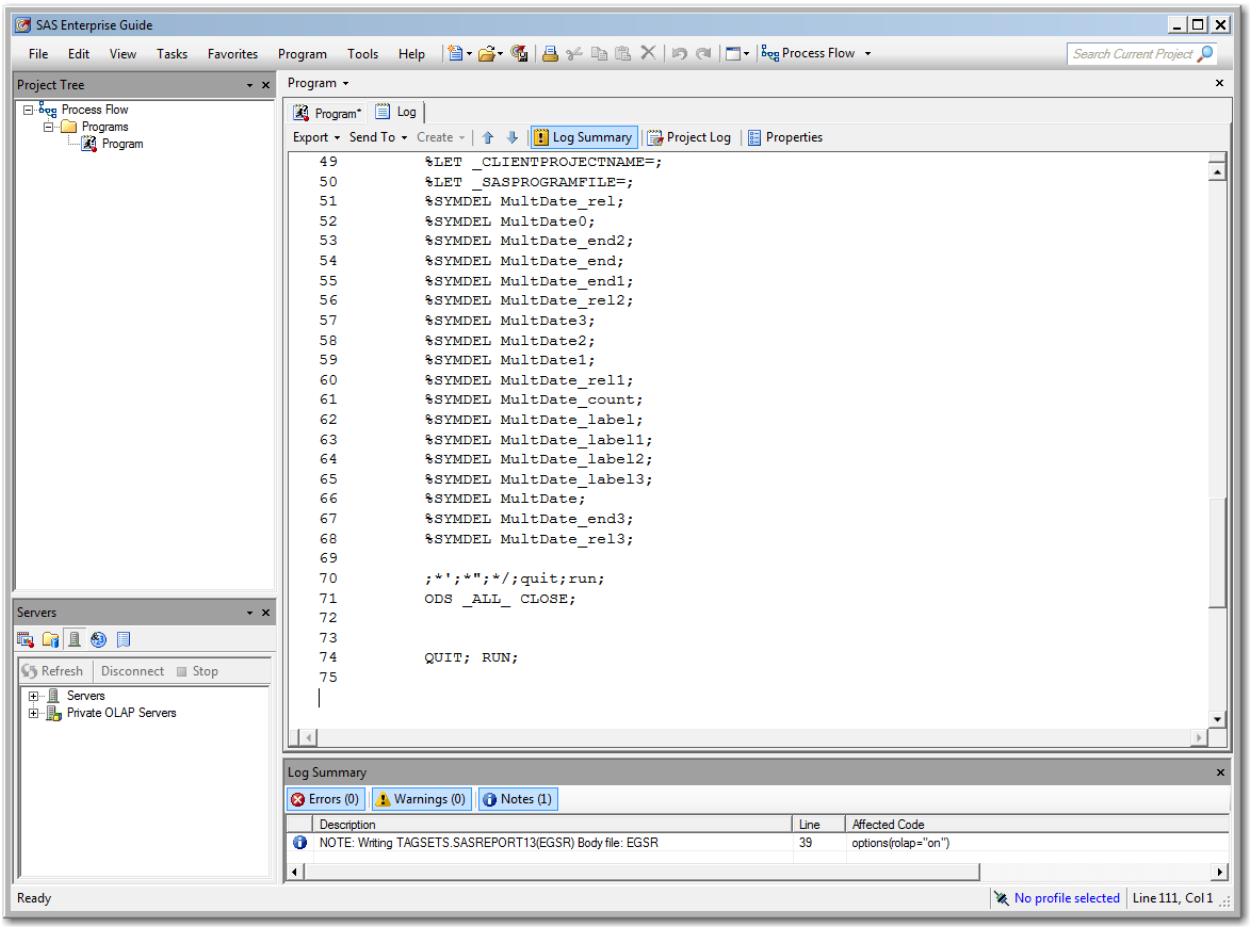
Display 167 - %LET Statements for Multiple Value Date Prompt

The log of the Program node using the prompt definition displays the value of the global variables created by the prompt.



Display 168 - Global Variables for Multiple Value Date Prompt

Because the **Use prompt value throughout project** is not checked for this prompt, the %SYMDEL statements remove the macro variables at the end of the program.



Display 169 - %SYMDEL Statements Remove MultDate* Macro Variables

SAS Studio

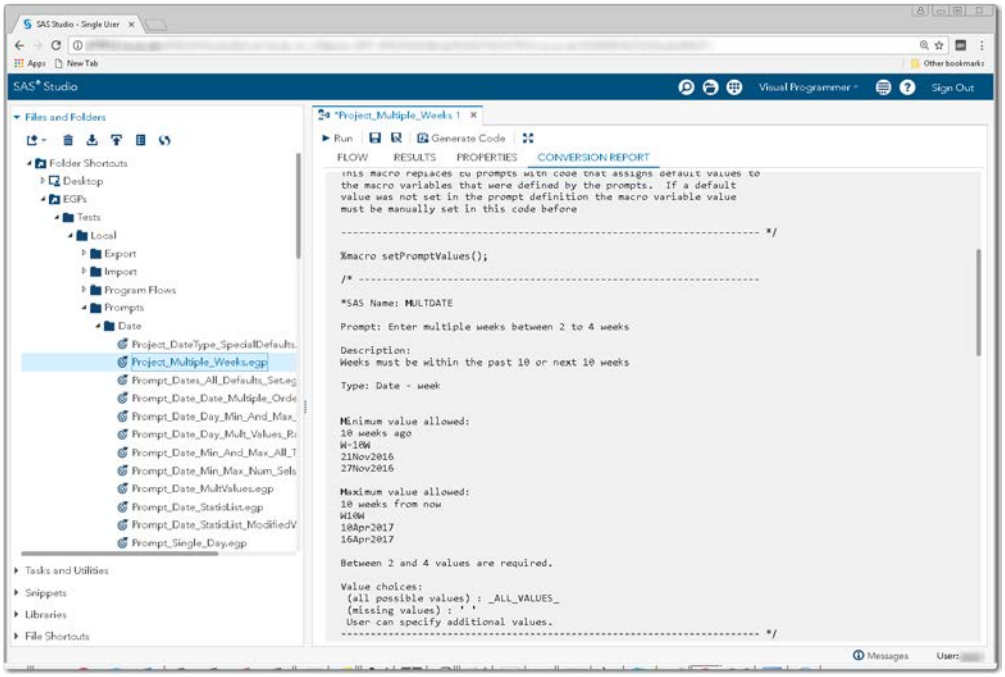
The following display shows the code that is added to the converted Program node for the multiple values date prompt in SAS Enterprise Guide.

These global date macro variables are created:

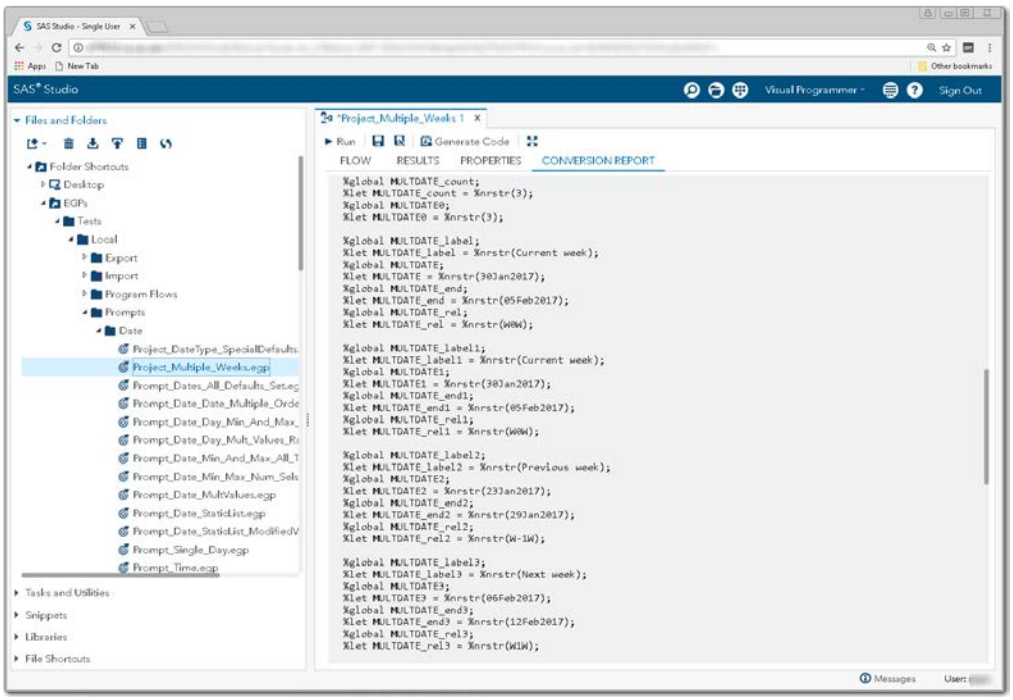
- MULTDATE
- MULTDATE_COUNT
- MULTDATE0
- MULTDATE1
- MULTDATE_label1
- MULTDATE_end1
- MULTDATE_rel1
- MULTDATE2
- MULTDATE_label2
- MULTDATE_end2
- MULTDATE_rel2
- MULTDATE3
- MULTDATE_label3
- MULTDATE_end3
- MULTDATE_rel3

The %LET statements assign the default values to MULTDATEn and MULTDATE_*n variables where n is 1-3 and * is label, end, or rel.

If you want to run your process flow against different input for the MULTDATE prompt, you must manually update the values of the macro variables in the %LET statements. Note that in this example, the MULTDATE_COUNT and MULTDATE_0 variables must reflect the number of text selections you want your program to process, and the MULTDATEn and MULTDATE_*n variables must be in sequential order.

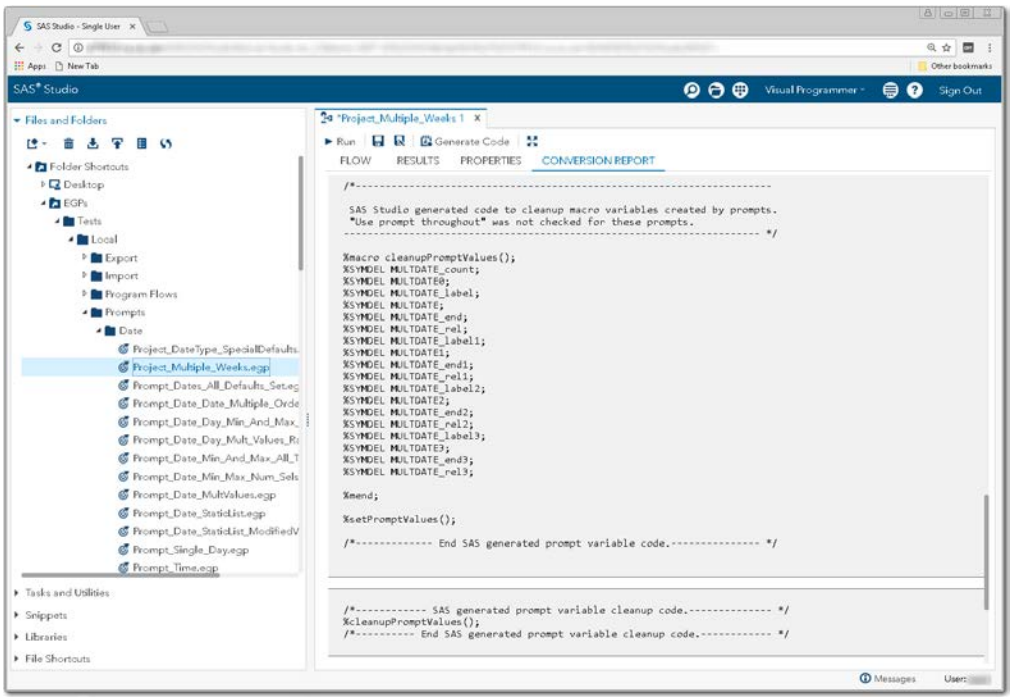


Display 170 - Code Comments for Multiple Date Prompt



Display 171 - Macro Variable Code for Multiple Date Prompt

Because the **Use prompt value throughout project** option is not checked, the %SYMDEL statements remove the MULTDATE* macro variables.

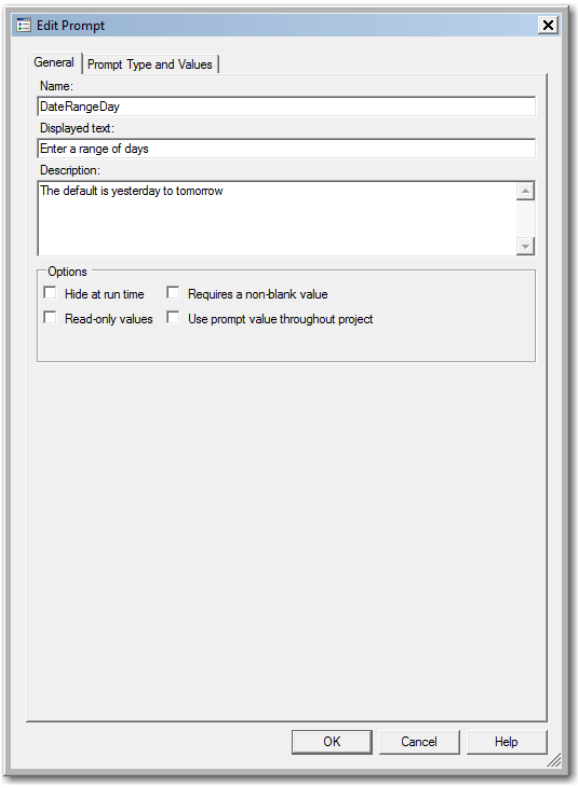


Display 172 - %SYMDEL Statements Remove the MULTDATE* Macro Variables

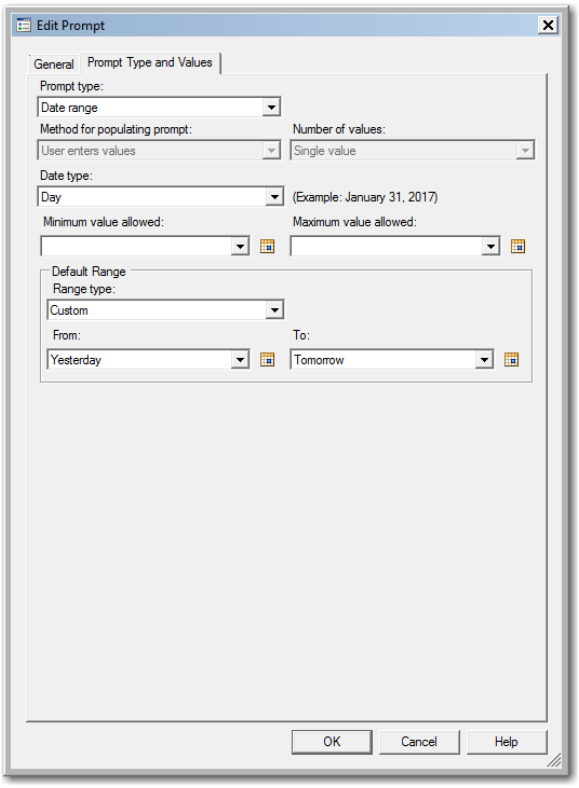
Date Range

SAS Enterprise Guide

In this example, a date range prompt named DateRangeDay is defined as shown in the following two displays.

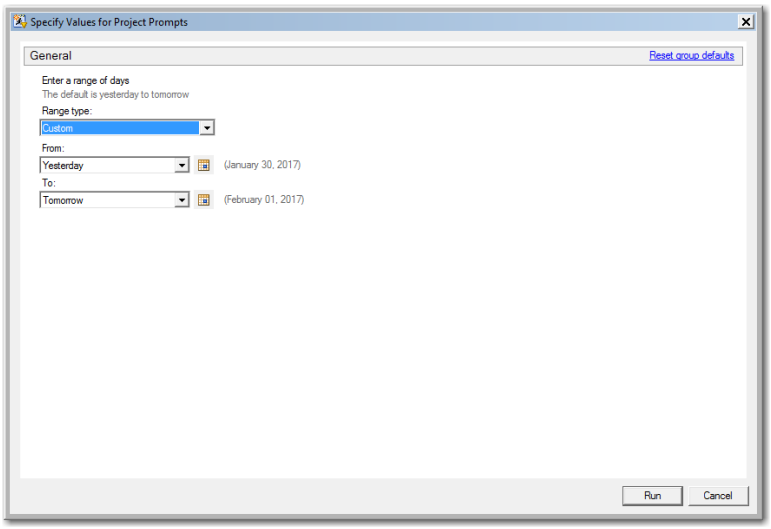


Display 173 - General Properties for Date Range Prompt



Display 174 - Type and Values for Date Range Prompt

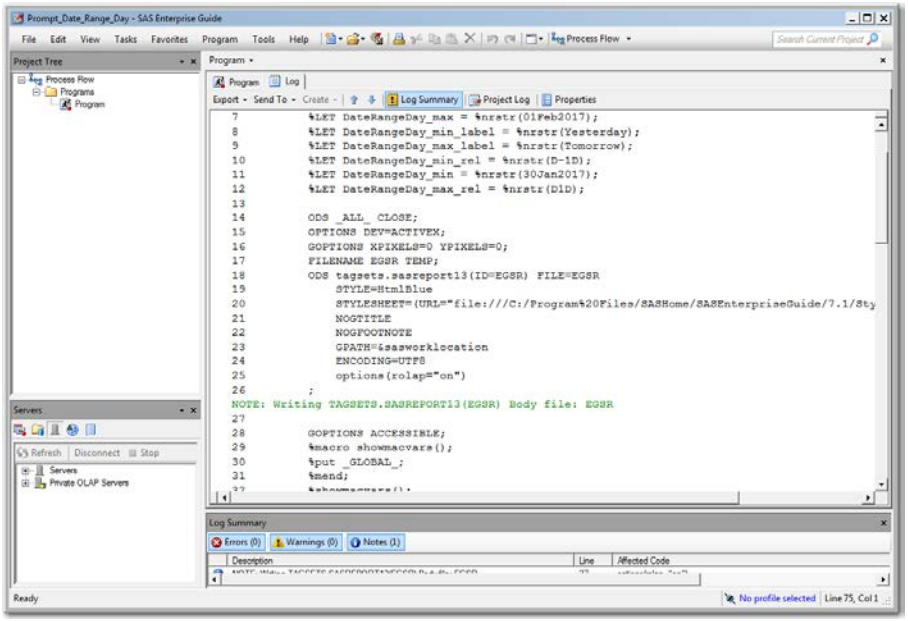
When you run the Program node that depends on the prompt, the following dialog box appears:



Display 175 - Date Range Prompt in Prompt Dialog Box

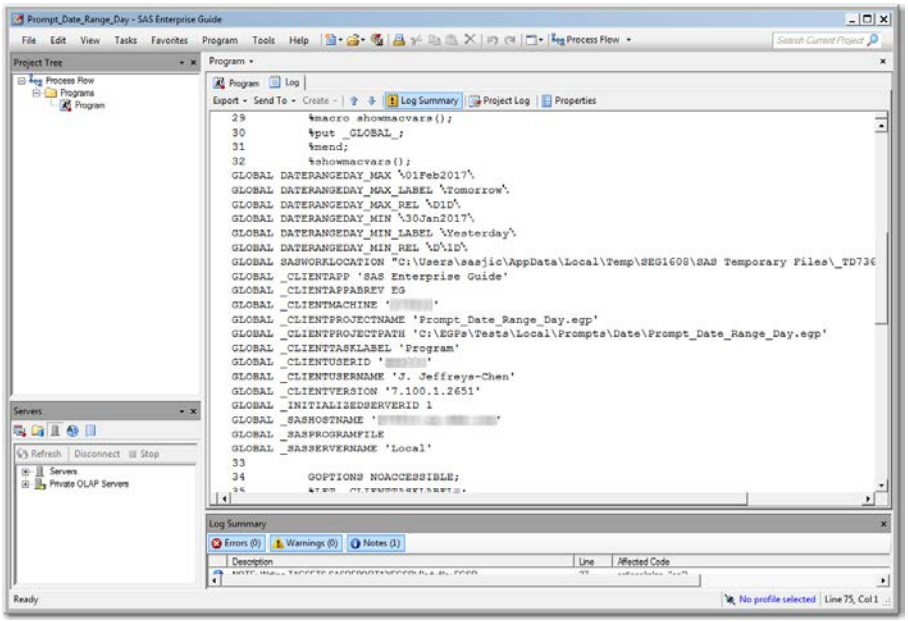
If the user leaves the default values in the date range value prompt fields, the following code is generated by SAS Enterprise Guide in the Program node that depends on the prompt.

The %LET statements assign the values specified in the prompt dialog box to the DateRangeDay* macro variables.



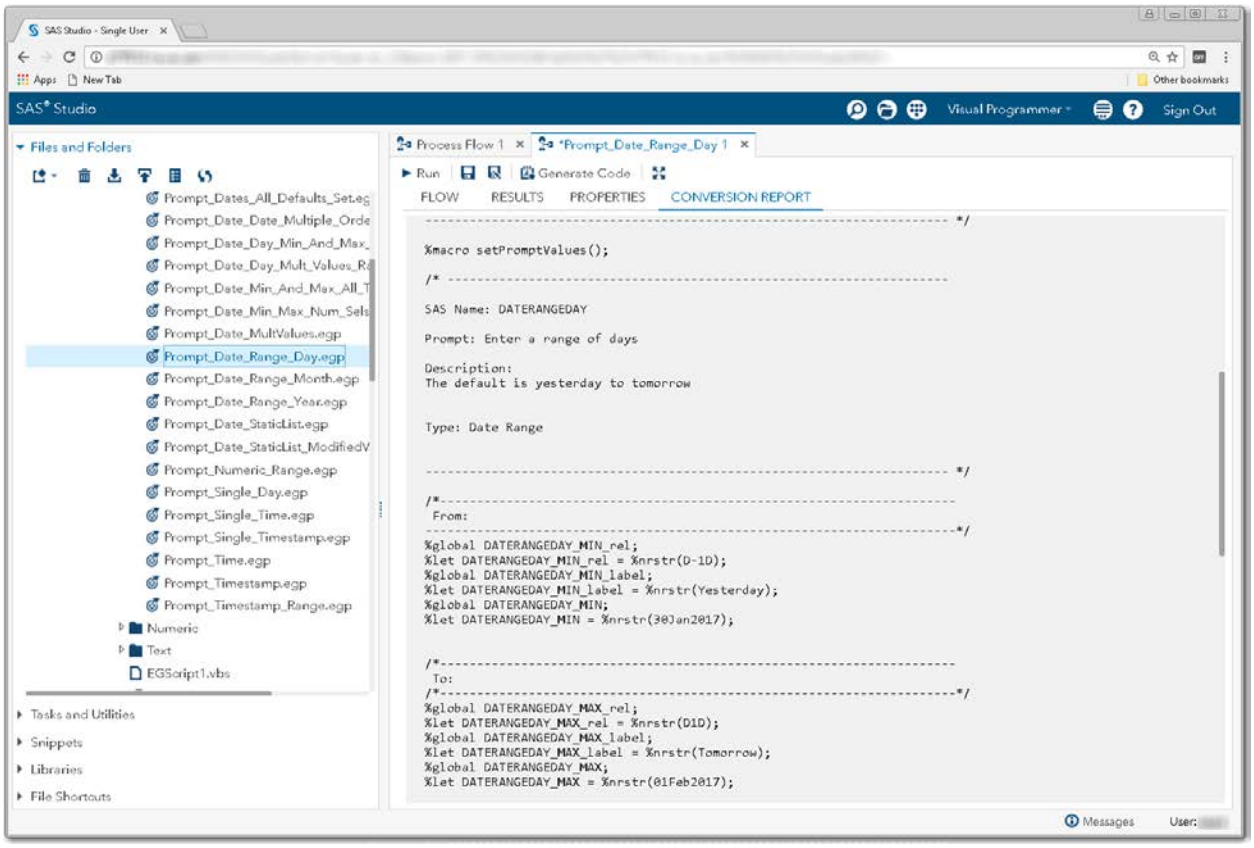
Display 176 - %LET Statements for Date Range Prompt

The log of the [Program node using the prompt definition] displays the values of the global variables created by the prompt.



Display 177 - Macro Varaibles for Date Range Prompt

Because the **Use prompt value throughout project** option is not checked for this prompt, the %SYMDEL statements remove the macro variables at the end of the program.



Display 178 - %SYMDEL Statements Remove DATERANGEDAY* Macro Variables

SAS Studio

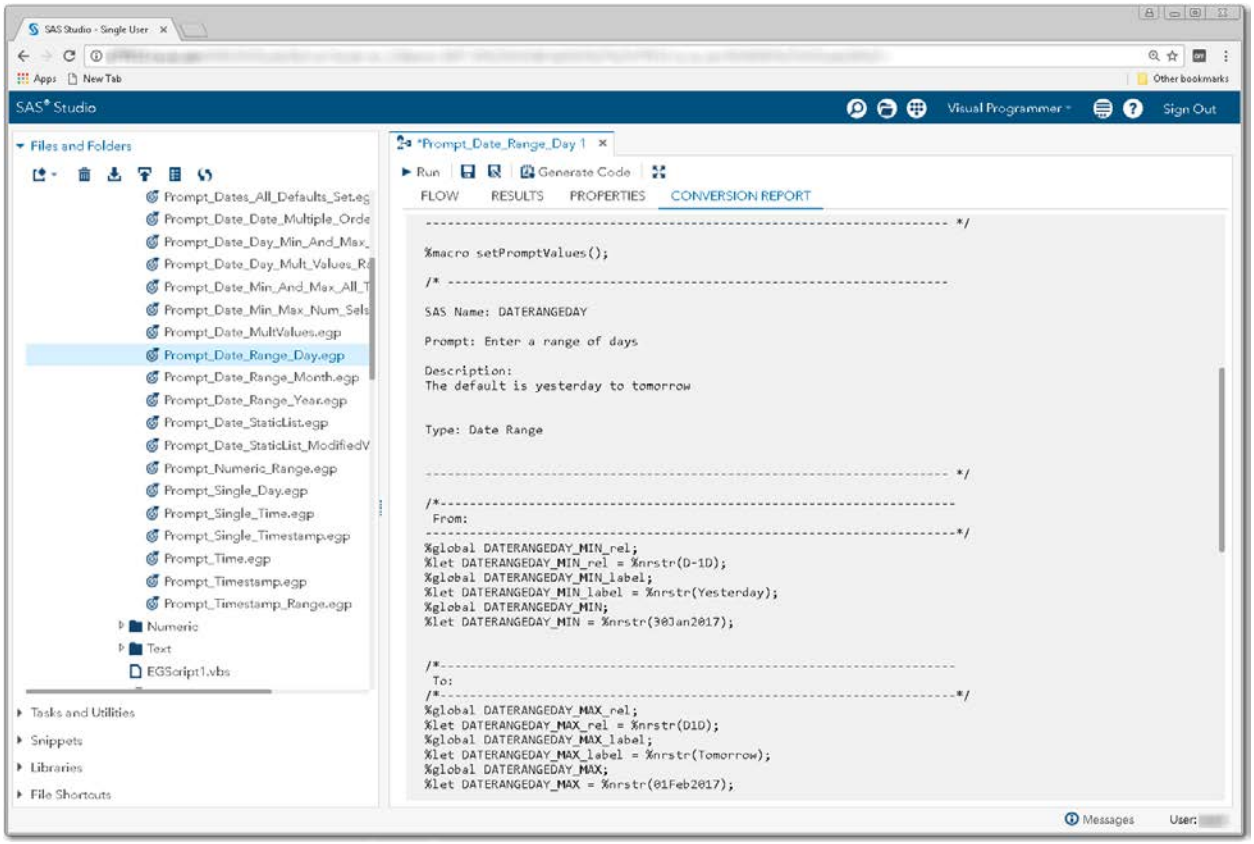
The following display shows the code that is added to the converted Program node for the date range prompt in SAS Enterprise Guide.

These global date macro variables are created:

- DATERANGEDAY_MIN_rel
- DATERANGEDAY_MIN_label
- DATERANGEDAY_MIN
- DATERANGEDAY_MAX_rel
- DATERANGEDAY_MAX_label
- DATERANGEDAY_MAX

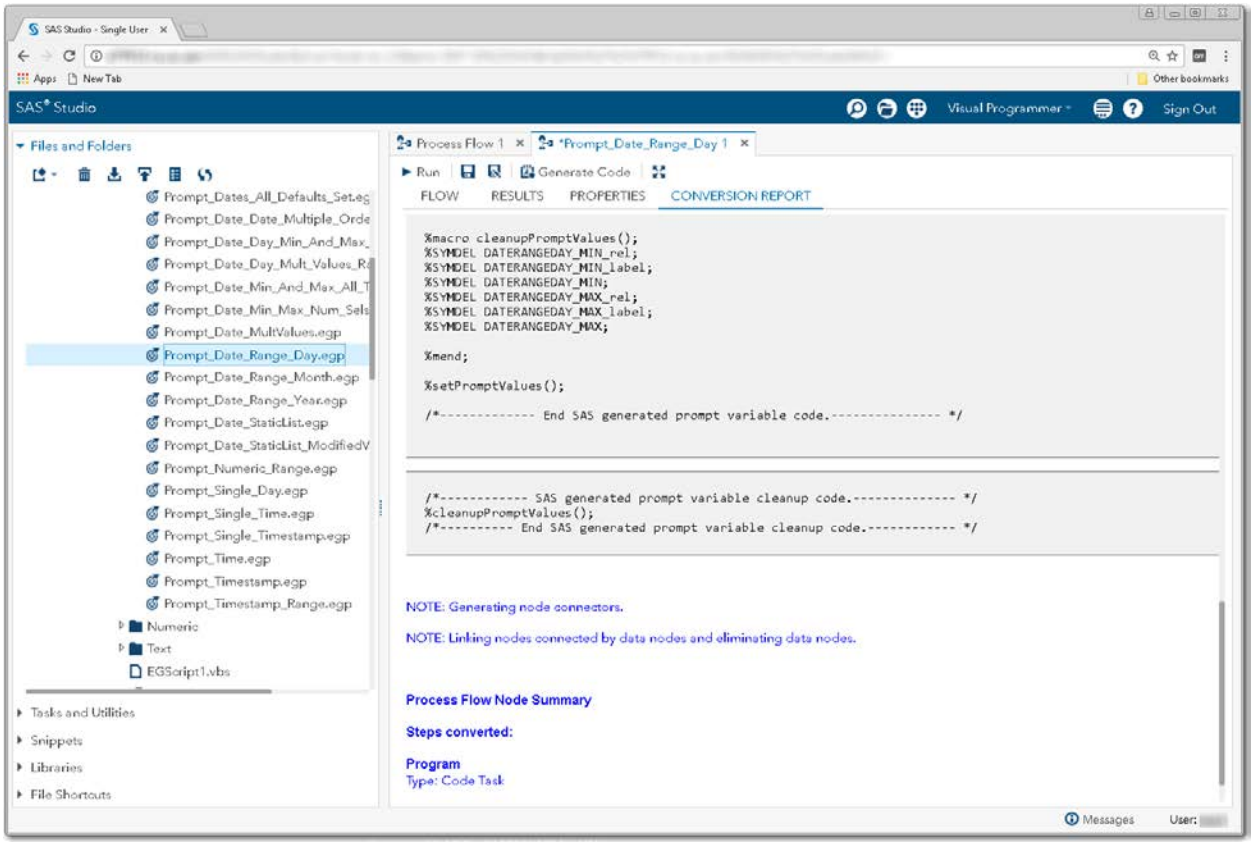
The %LET statements assign the default values to the DATERANGEDAY macro variables.

If you want to run your process flow using different values for the DATERANGEDAY prompt, you must manually update values of the macro variables in the %LET statements.



Display 179 - Macro Code for Date Range Values

Because the **Use prompt value throughout project** option is not checked, the %SYMDEL statements remove the DATERANGEDAY* macro variables.



Display 180 - %SYMDEL Statements Remove DATERANGEDAY* Macro Variables

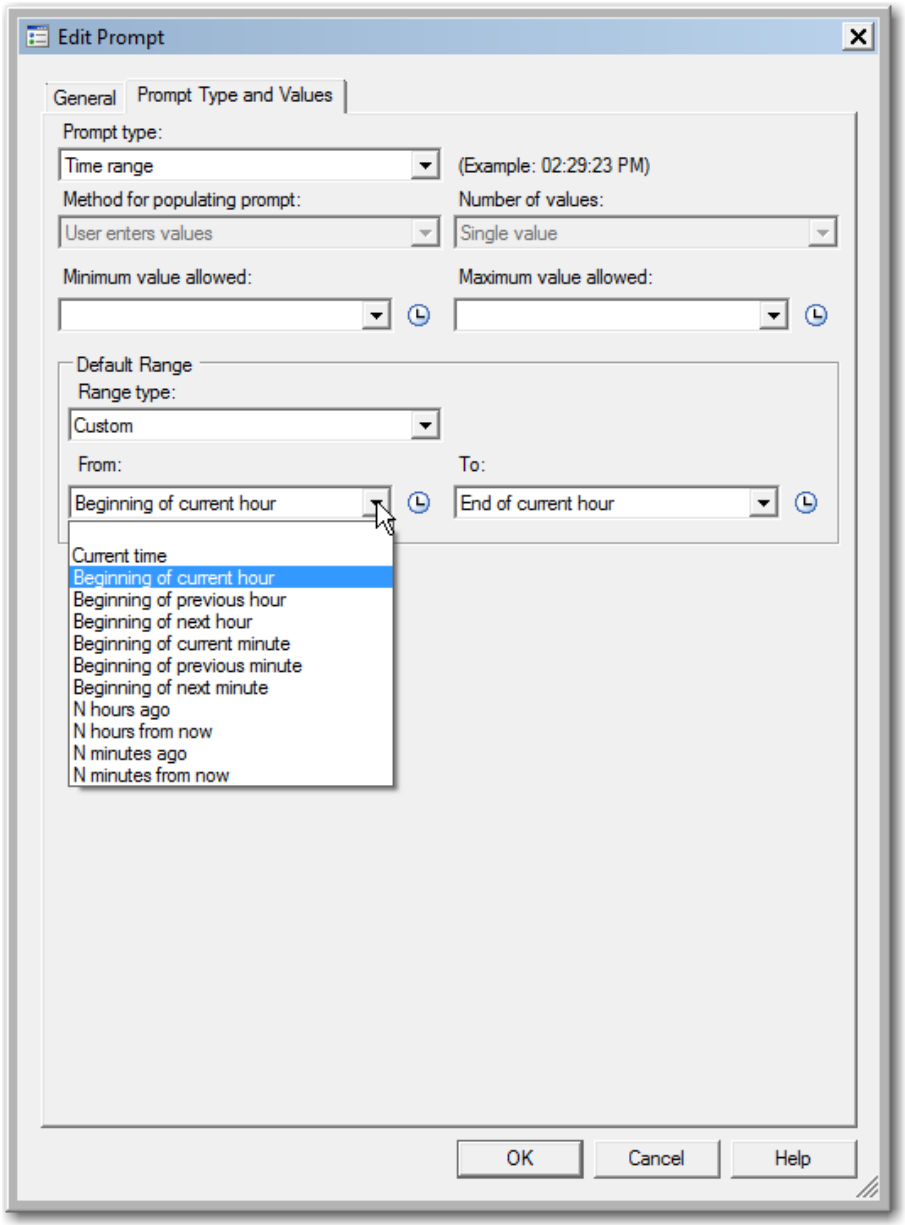
Note: For a Date Range prompt with a period of Month or Year, the macro code does not generate correctly in SAS Studio due to a bug.

Time

Time and Timestamp Macro Variable Explanation

Macro variables generated by SAS Enterprise Guide for time prompts include the basic prompt name, name_rel, and name_label.

- The name macro variable will contain the absolute time specified or the exact time of the common value, for example 15:46:33.
- The name_rel macro variable will contain a code for a common value. For example, “t0m” represents “Current Time”.
- The name_label macro variable will contain the common value, such as “Current Time” or “Beginning of Current Hour”.

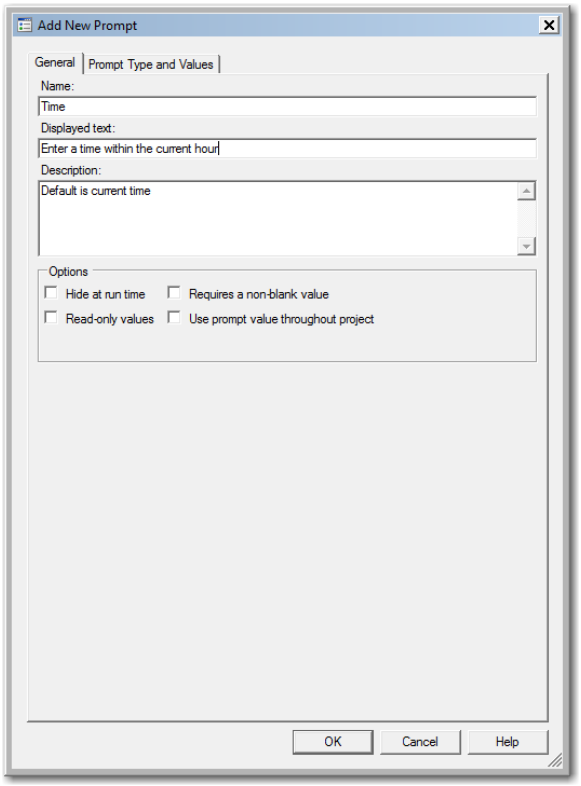


Display 181 – Common Values for Time

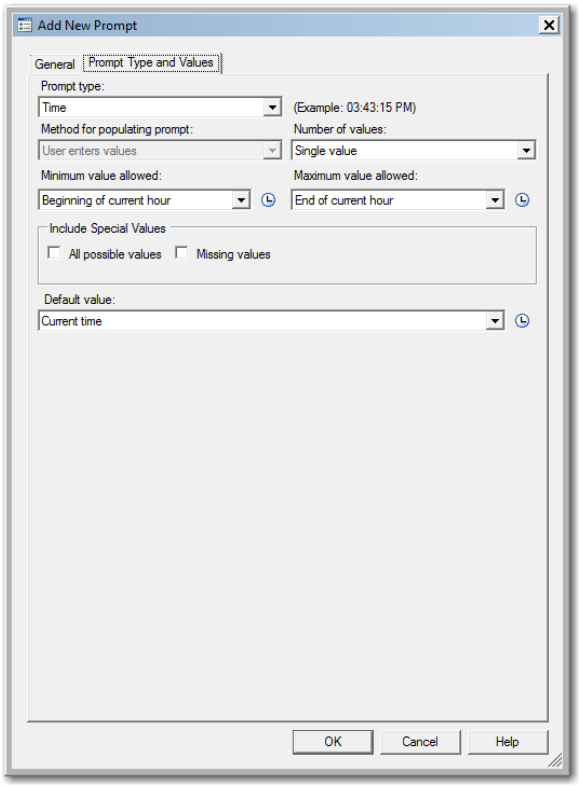
Single Time

SAS Enterprise Guide

In this example, a single value time prompt named Time is defined as shown in the following two displays.

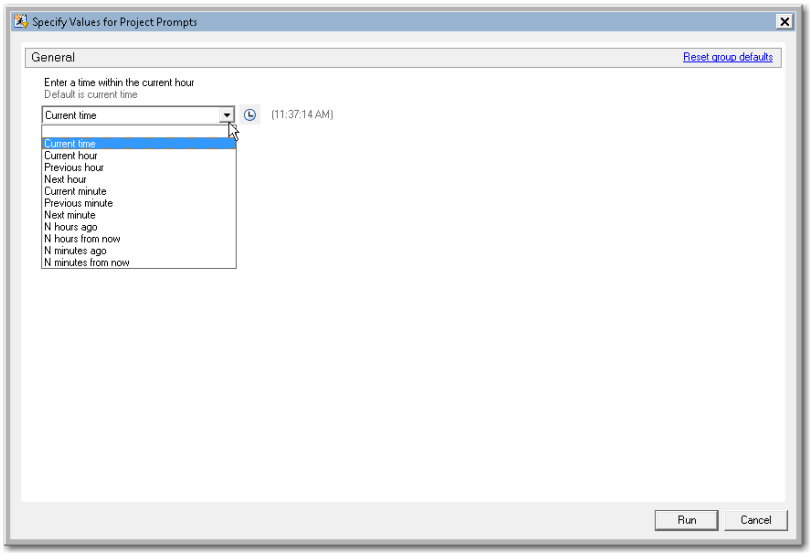


Display 182 - General Properties for Time Prompt



Display 183 - Type and Values for Time Prompt

When you run the Program node that depends on the prompt, the following dialog box appears.

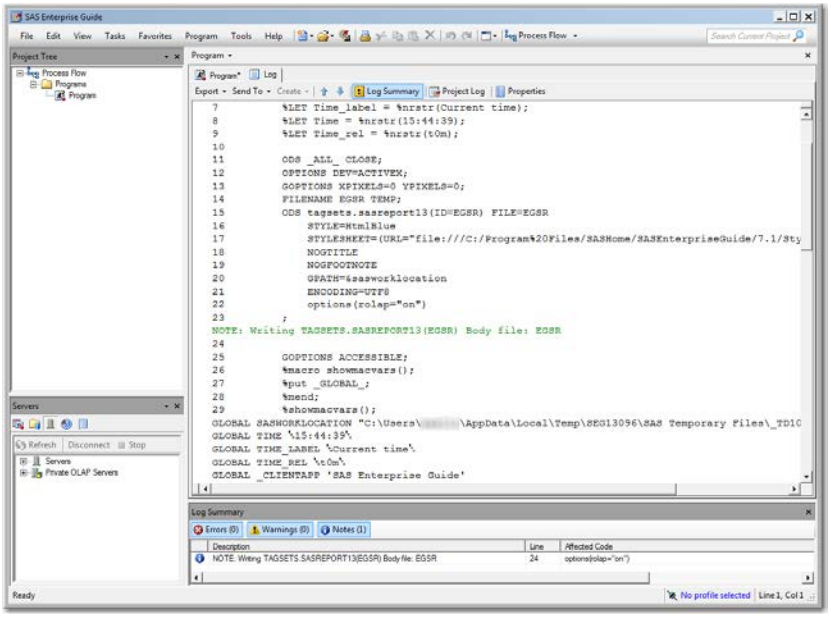


Display 184 - Time Prompt in Prompt Dialog Box

If the user leaves the default value in the time prompt field, the following code is generated by SAS Enterprise Guide in the Program node that depends on the prompt.

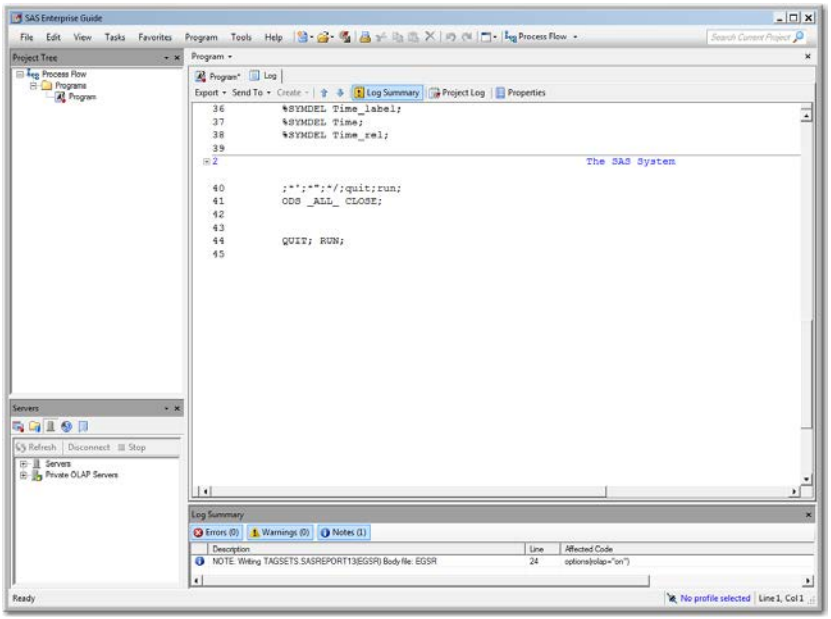
The %LET statements assign the value specified in the prompt dialog box to the macro Time* variables.

The log of the Program node using the prompt definition displays the values of the global variables created by the prompt.



Display 185 - Global Macro Variables and %LET Statements for Time Prompt

Because the **Use prompt value throughout project** option is not checked for this prompt, the %SYMDEL statements remove the macro variables at the end of the program.



Display 186 - %SYMDEL Statements Remove Time* Macro Variables

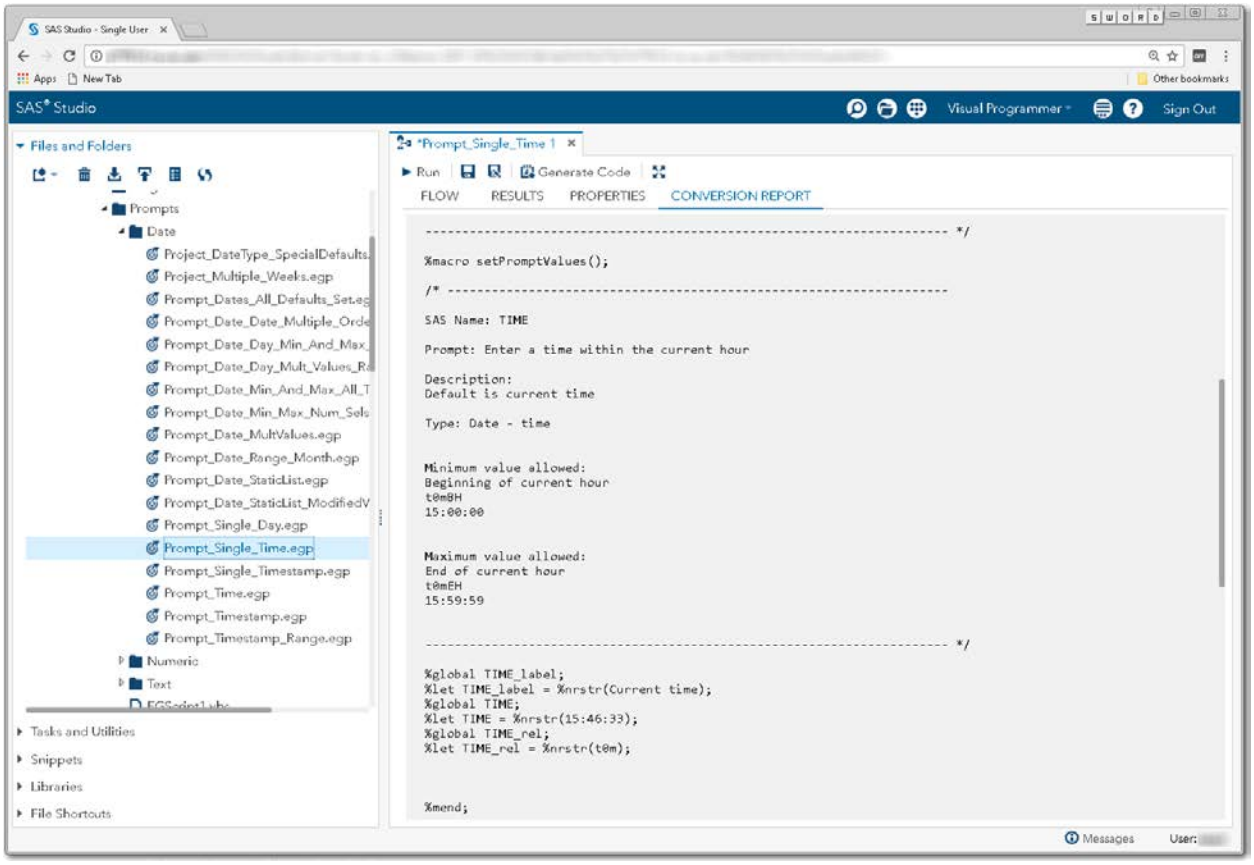
SAS Studio

The following display shows the code that is added to the converted Program node for the time prompt in SAS Enterprise Guide. These global time macro variables are created:

- TIME_rel
- TIME_label
- TIME

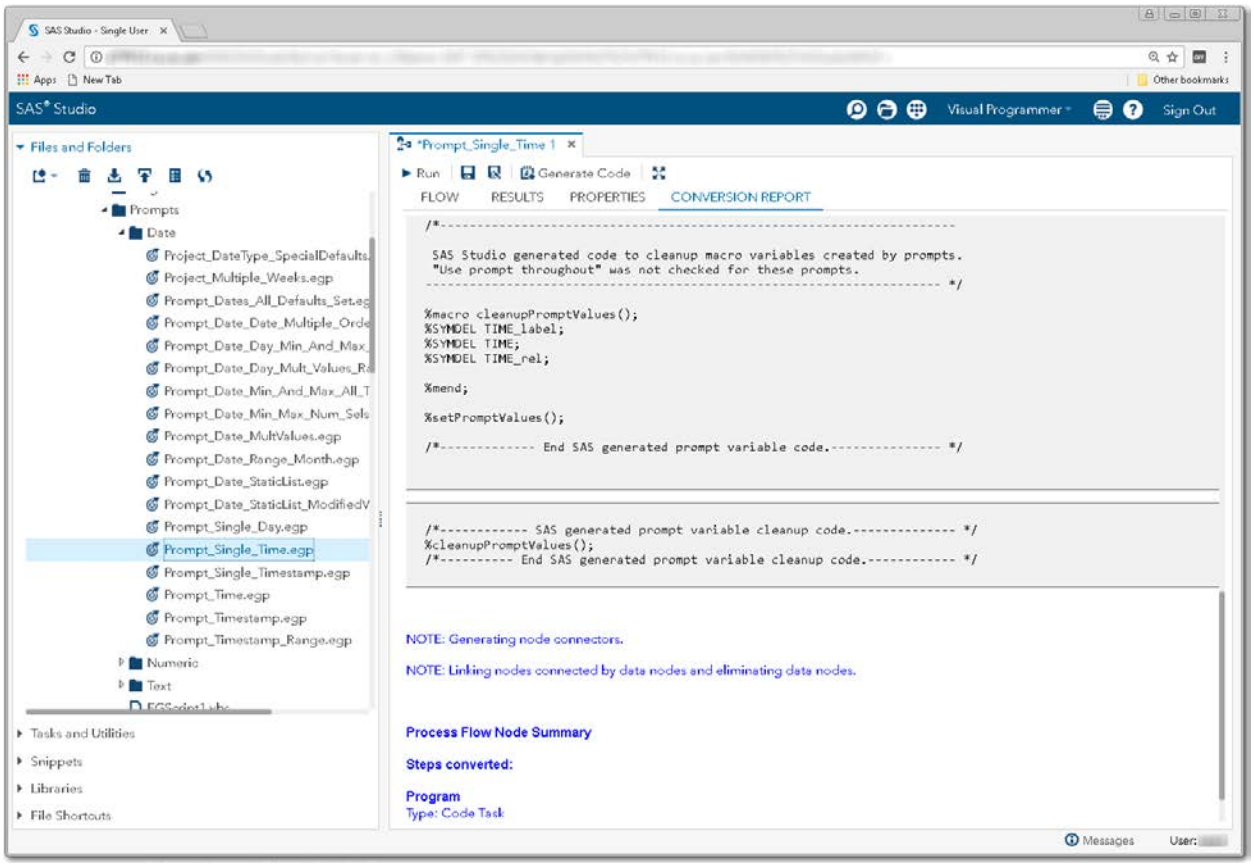
The %LET statements assign the default values to the TIME macro variables.

If you want to run your process flow using different values for the TIME prompt, you must manually update the values of the macro variables in the %LET statements.



Display 187 - Macro Code for Time Prompt

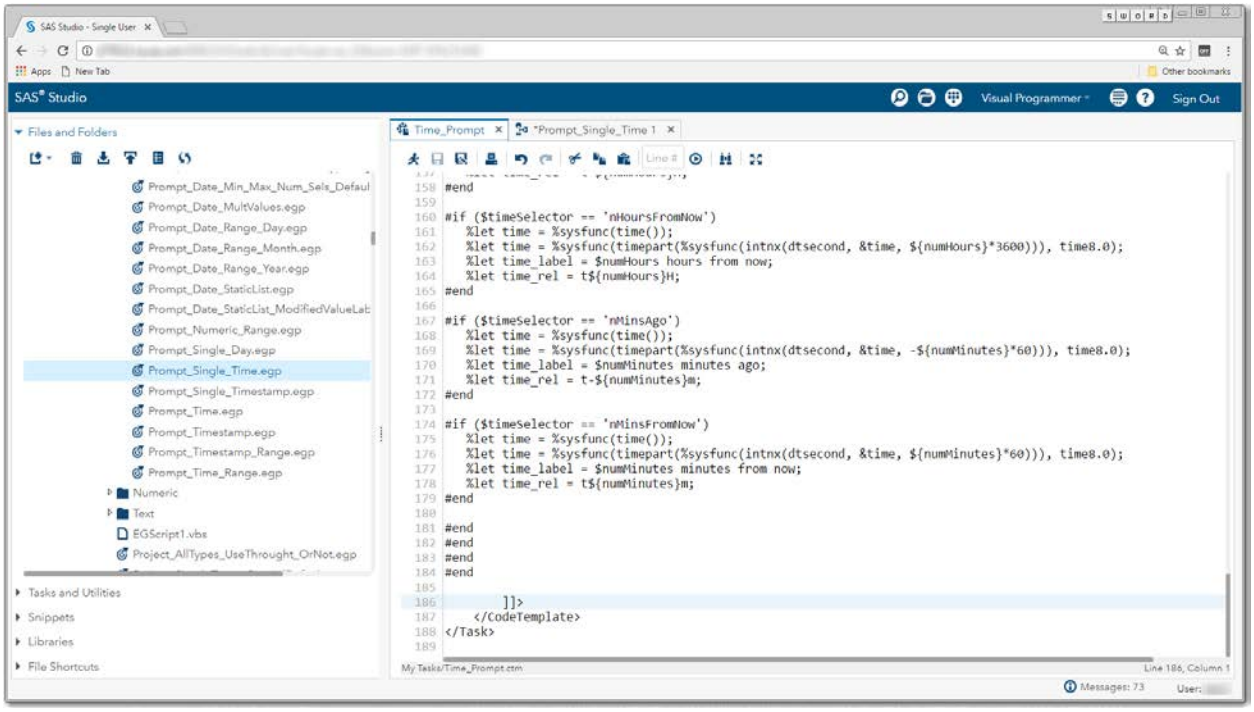
Because the **Use prompt value throughout project** option is not checked, the %SYMDEL statements remove the TIME macro variables.



Display 188 - %SYMDEL Statements Remove TIME* Macro Variables

Substituting a SAS Studio Task for Time Prompt

1. Create a SAS Studio task with a control that represents the Time prompt.
   - Add controls as shown in the Time_Prompt task.
   - Set the default value to the default value shown in the generated `setPromptValues()` macro in the converted Program node.
   - Change the strings for the input controls to match the strings specified in the prompt.



Display 189 - Replacement Task for Time Prompt

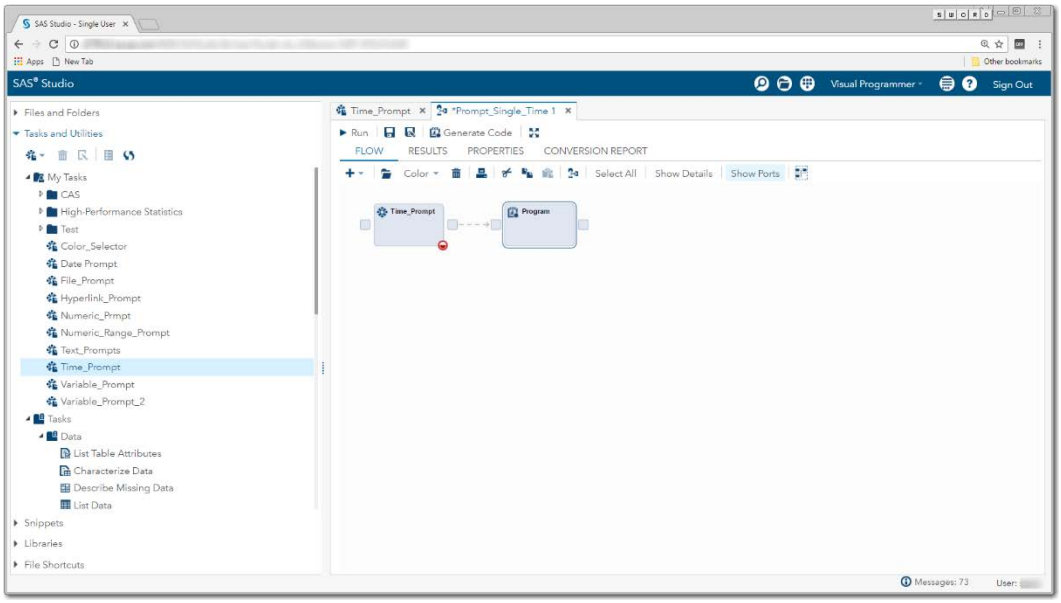
The following code is an example of a task that could be used as the time prompt.

```
<?xml version="1.0" encoding="UTF-8"?><Task runNLS="never" schemaVersion="5.1">
       <Registration>
       <Name>Time_Prompt</Name>
       <Description>Time Prompt</Description>
       <GUID/>
       <Procedures>TBD</Procedures>
       <Version>3.6</Version>
       <Links></Links>
       </Registration>
       <Metadata>
              <DataSources></DataSources>
              <Options>
                  <Option inputType="string" name="label">
                      Enter a time within the current hour
                   </Option>
                  <Option defaultValue="currentTime" inputType="combobox"
                         name="timeSelector">
                       Default is current time
                   </Option>
                   <Option inputType="string" name="specify">Specify time</Option>
                   <Option inputType="string" name="currentTime">Current time</Option>
                   <Option inputType="string" name="currentHour">Current hour</Option>
                   <Option inputType="string" name="prevHour">Previous hour</Option>
                   <Option inputType="string" name="nextHour">Next hour</Option>
                   <Option inputType="string" name="currMin">Current minute</Option>
                   <Option inputType="string" name="prevMin">Previous minute</Option>
                   <Option inputType="string" name="nextMin">Next minute</Option>
                   <Option inputType="string" name="nHoursAgo">N hours ago</Option>
                  <Option inputType="string" name="nHoursFromNow">
                      N hours from now
                  </Option>
                  <Option inputType="string" name="nMinsAgo">
                      N minutes ago
                  </Option>
                  <Option inputType="string" name="nMinsFromNow">
                      N minutes from now
                  </Option>
                  <Option inputType="inputtext" name="timeInput"
                          promptMessage="Example: 01:23:45 PM" />
                  <Option defaultValue="N" inputType="numbertext"
                          invalidMessage="Invalid value. Enter a positive integer."
                          minValue="0"
                          missingMessage="Enter number of minutes"
                          name="numMinutes"
                          promptMessage="Enter number of minutes"
                          rangeMessage=
                             "This number is out of range.  Enter a positive number.">
                        Number of minutes:
                  </Option>
                  <Option defaultValue="N" inputType="numbertext"
                          invalidMessage="Invalid value. Enter a positive integer."
                          minValue="0" missingMessage="Enter number of hours"
                          name="numHours"
                          promptMessage="Enter number of hours"
                          rangeMessage=
                             "This number is out of range.  Enter a positive number.">
                     Number of hours:
                  </Option>
              </Options>
       </Metadata>
```

```
      <UI>
           <OptionItem option="label"/>
             <OptionChoice option="timeSelector">
                   <OptionItem option="specify"/>
                   <OptionItem option="currentTime"/>
                   <OptionItem option="currentHour"/>
                   <OptionItem option="prevHour"/>
                   <OptionItem option="nextHour"/>
                   <OptionItem option="currMin"/>
                   <OptionItem option="prevMin"/>
                   <OptionItem option="nextMin"/>
                   <OptionItem option="nHoursAgo"/>
                   <OptionItem option="nHoursFromNow"/>
                   <OptionItem option="nMinsAgo"/>
                   <OptionItem option="nMinsFromNow"/>
             </OptionChoice>

             <OptionItem option="timeInput"/>
             <OptionItem option="numMinutes"/>
             <OptionItem option="numHours"/>
      </UI>
      <Dependencies>
            <Dependency condition="($timeSelector == 'specify')">
               <Target action="hide" conditionResult="false" option="timeInput"/>
               <Target action="show" conditionResult="true" option="timeInput"/>
            </Dependency>
            <Dependency condition=
               "(($timeSelector == 'nMinsAgo') || ($timeSelector == 'nMinsFromNow'))">
               <Target action="hide" conditionResult="false" option="numMinutes"/>
               <Target action="show" conditionResult="true" option="numMinutes"/>
            </Dependency>
            <Dependency condition=
             "(($timeSelector == 'nHoursAgo') || ($timeSelector == 'nHoursFromNow'))">
               <Target action="hide" conditionResult="false" option="numHours"/>
               <Target action="show" conditionResult="true" option="numHours"/>
             </Dependency>
        </Dependencies>
       <CodeTemplate>
             <![CDATA[
%global time;
%global time_rel;
%global time_label;

#if ($timeSelector == 'specify')
   %let time = $timeInput;
   %let time = %sysfunc(inputn(&time, TIME10.),TOD);
   %let time_label = $timeInput;
   %symdel time_rel;
#else
#if ($timeSelector == 'currentTime')
   %let time = %sysfunc(time(),time8.0);
   %let time_label = Current time;
   %let time_rel = t0m;
#else
#if ($timeSelector == 'currentHour')
   %let time = %sysfunc(time());
   %let time = %sysfunc(intnx(hour, &time, 0, b), time8.0);
   %let time_label = Current hour;
   %let time_rel = H0H;
#else
#if ($timeSelector == 'currMin')
   %let time = %sysfunc(time());
   %let time = %sysfunc(intnx(minute, &time, 0, b), time8.0);
```

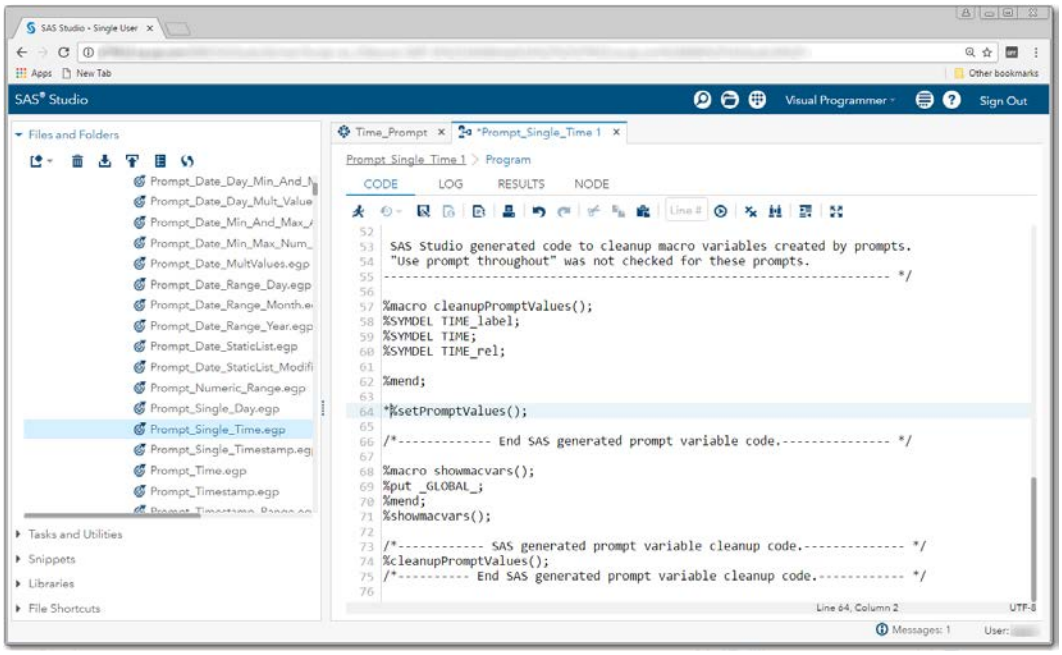
```
   %let time_label = Current minute;
   %let time_rel = mom;
#else
#if ($timeSelector == 'prevHour')
   %let time = %sysfunc(time());
   %let time = %sysfunc(timepart(%sysfunc(intnx(dthour, &time, -1))), time8.0);
   %let time_label = Previous hour;
   %let time_rel = H-1H;
#end
#if ($timeSelector == 'nextHour')
   %let time = %sysfunc(time());
   %let time = %sysfunc(timepart(%sysfunc(intnx(dthour, &time, 1))), time8.0);
   %let time_label = Next hour;
   %let time_rel = H1H;
#end
#if ($timeSelector == 'prevMin')
   %let time = %sysfunc(time());
   %let time = %sysfunc(timepart(%sysfunc(intnx(dtminute, &time, -1))), time8.0);
   %let time_label = Previous minute;
   %let time_rel = m-1m;
#end
#if ($timeSelector == 'nextMin')
   %let time = %sysfunc(time());
   %let time = %sysfunc(timepart(%sysfunc(intnx(dtminute, &time, 1))), time8.0);
   %let time_label = Next minute;
   %let time_rel = m1m;
#end
#if ($timeSelector == 'nHoursAgo')
   %let time = %sysfunc(time());
   %let time = %sysfunc(timepart(%sysfunc(intnx(dtsecond, &time, -${numHours}*3600))),
                                                  time8.0);
   %let time_label = $numHours hours ago;
   %let time_rel = t-${numHours}H;
#end
#if ($timeSelector == 'nHoursFromNow')
   %let time = %sysfunc(time());
   %let time = %sysfunc(timepart(%sysfunc(intnx(dtsecond, &time, ${numHours}*3600))),
                                                  time8.0);
   %let time_label = $numHours hours from now;
   %let time_rel = t${numHours}H;
#end

#if ($timeSelector == 'nMinsAgo')
   %let time = %sysfunc(time());
   %let time = %sysfunc(timepart(%sysfunc(intnx(dtsecond, &time, -${numMinutes}*60))),
time8.0);
   %let time_label = $numMinutes minutes ago;
   %let time_rel = t-${numMinutes}m;
#end
#if ($timeSelector == 'nMinsFromNow')
   %let time = %sysfunc(time());
   %let time = %sysfunc(timepart(%sysfunc(intnx(dtsecond, &time, ${numMinutes}*60))),
time8.0);
   %let time_label = $numMinutes minutes from now;
   %let time_rel = t${numMinutes}m;
#end
#end
#end
#end
#end
            ]]>
      </CodeTemplate>
</Task>
```

2. Save the prompt replacement task to your **My Tasks** folder.
3. Drag the task from **My Tasks** into your converted process flow.
4. Link the output port of the task to the input port of the converted Program node.



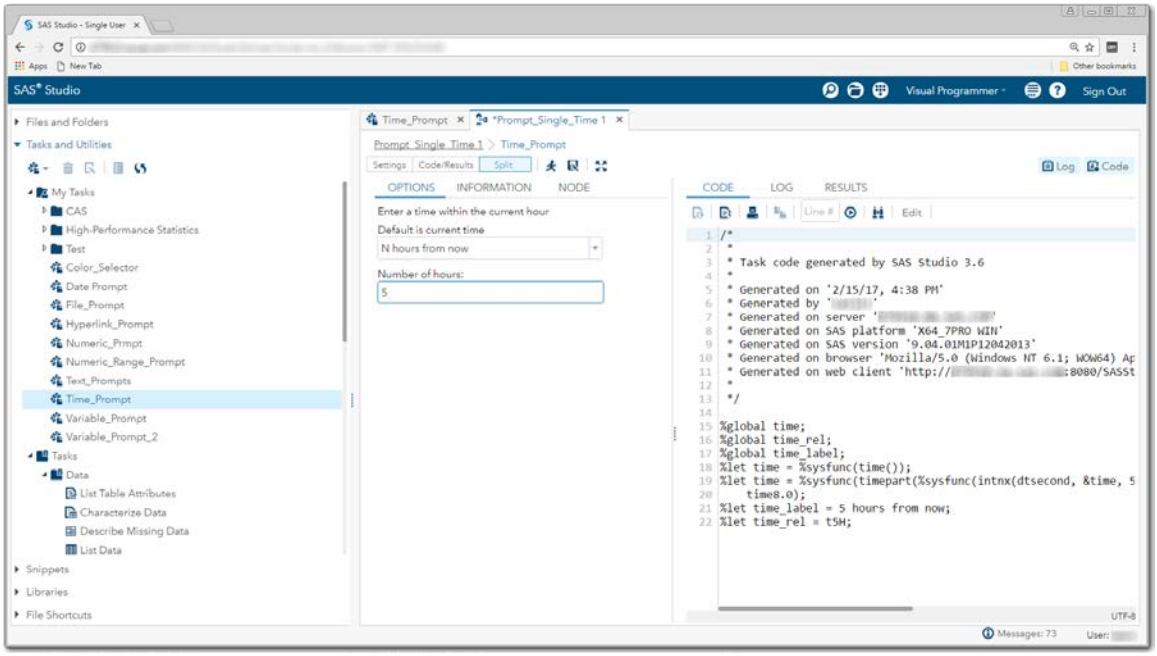
Display 190 - Time Input Task Linked to Program Node

5. Comment out the `%setPromptValues()` macro call from the converted Program node. The macro code generated by the time input task replaces this code.



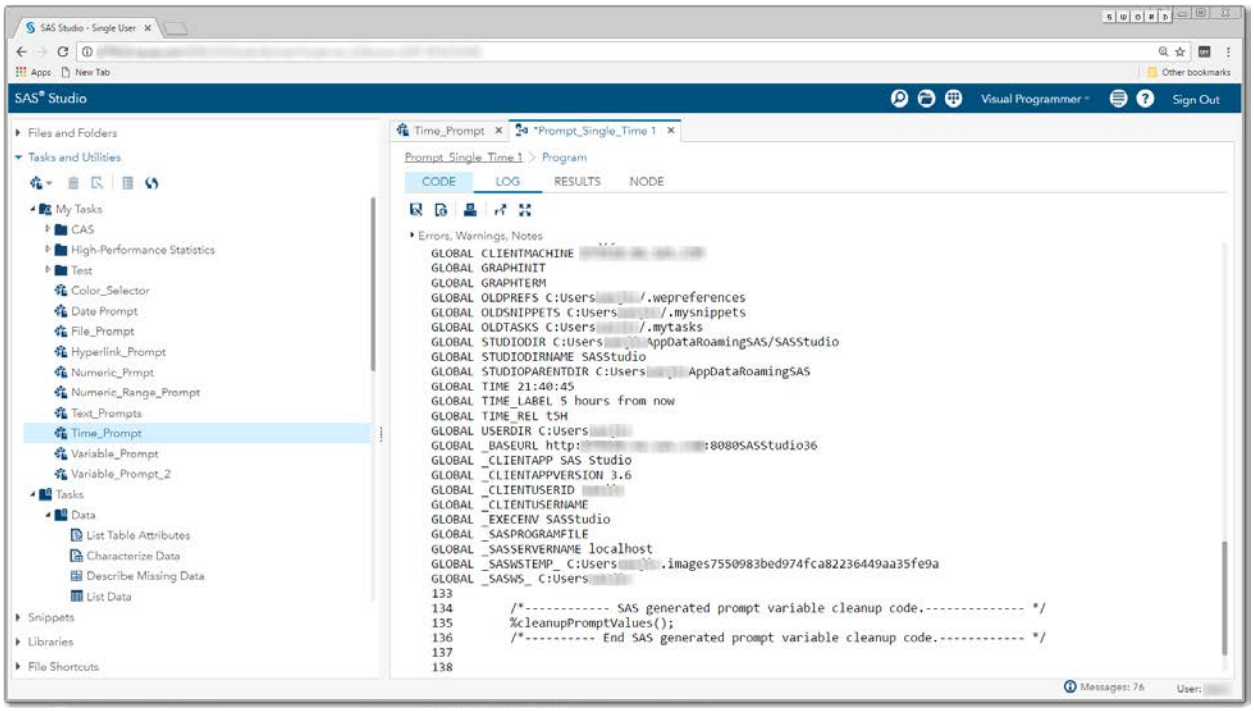
Display 191 – Commented out %setPromptValues in Program Node

To run your flow with a different time value than the default value, open the Time_Prompt node and specify a different time.



Display 192 – Time_Prompt Task

When you run the process flow, the global Time* variables are set to the value specified in the task.

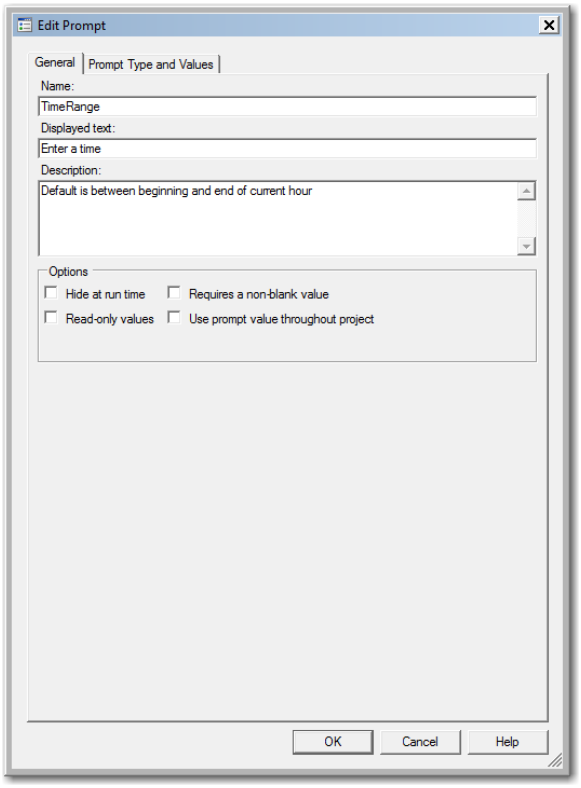


Display 193 – Time Prompt Variables with Updated Values

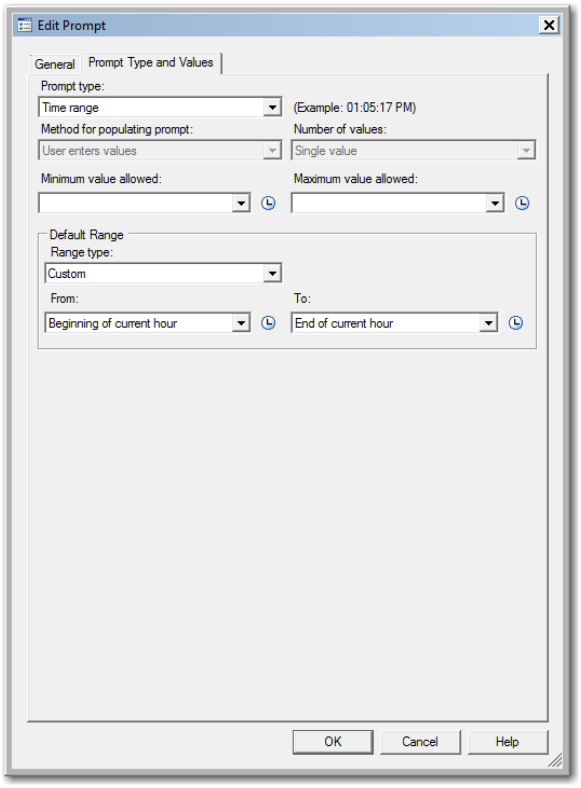
Time Range

SAS Enterprise Guide

In this example, a time range prompt named TimeRange is defined as shown in the following two displays.

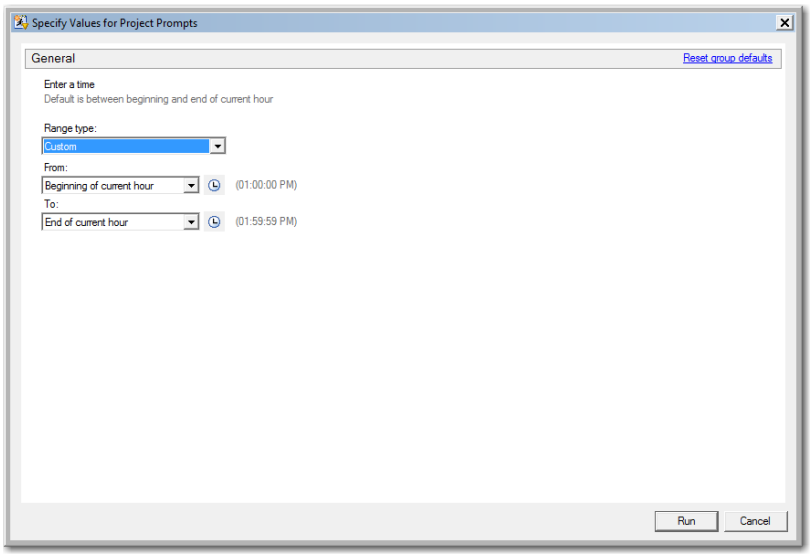


Display 194 - General Properties for Time Range Prompt



Display 195 - Type and Values for Time Range Prompt

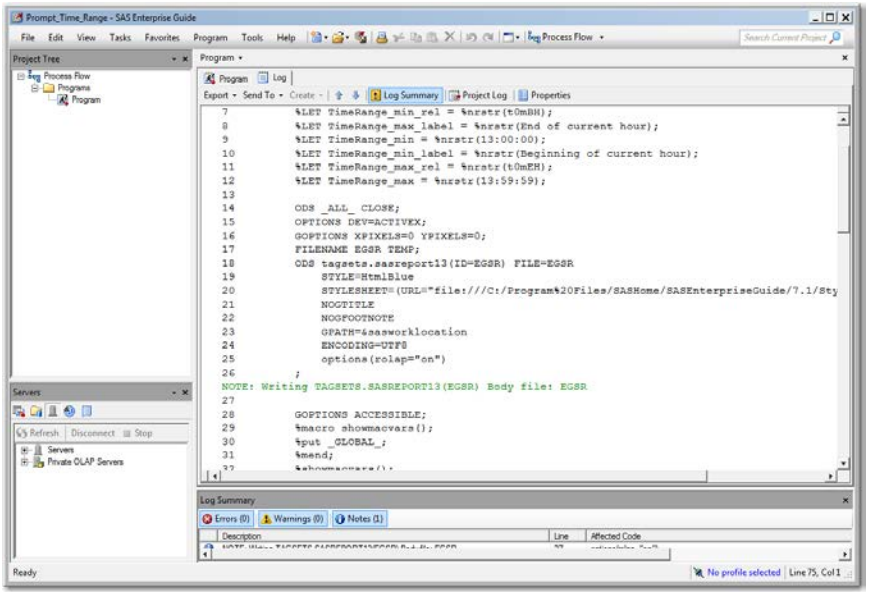
When you run the Program node that depends on the prompt, the following dialog box appears:



Display 196 - Time Range Prompt in Prompt Dialog Box

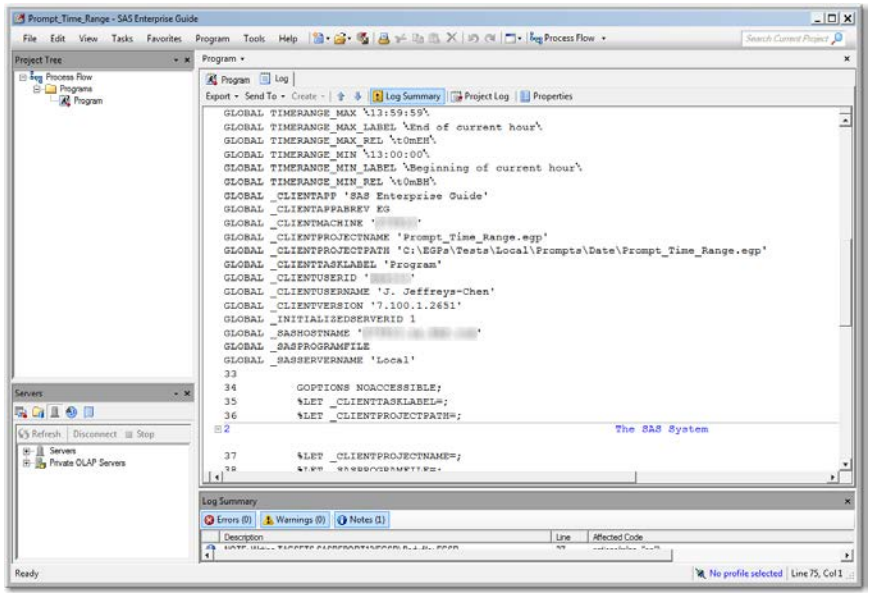
If the user leaves the default values in the prompt fields, the following code is generated by SAS Enterprise Guide in the Program node that depends on the prompt.

The %LET statements assign the values specified in the prompt dialog box to the Timerange* macro variables.



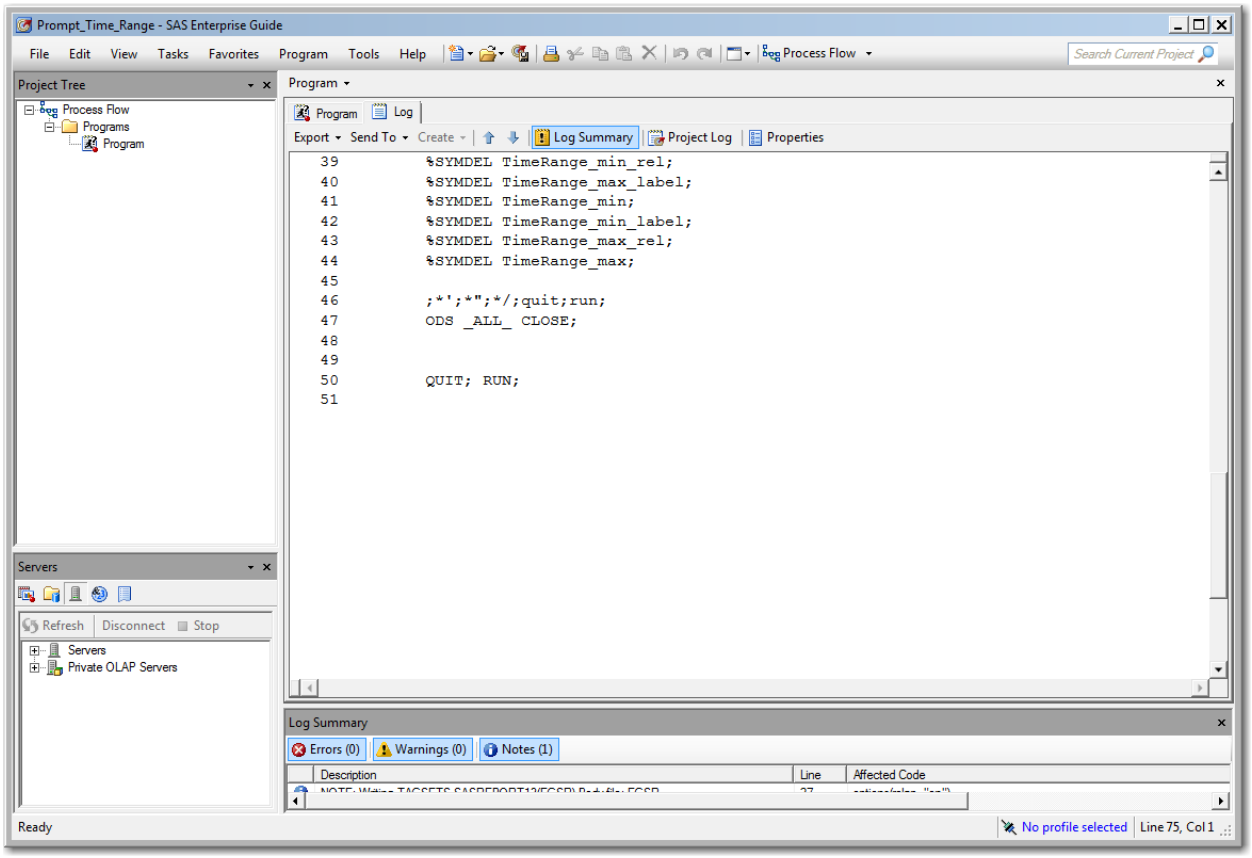
Display 197 - %LET Statements for Macro Variables of Time Range Prompt

The log of the Program node using the prompt definition displays the values of the global variables created by the prompt.



Display 198 – Global Macro Variables for Time Range Prompt

Because the **Use prompt value throughout project** option is not checked for this prompt, the %SYMDEL statements remove the macro variables at the end of the program.



Display 199 - %SYMDEL Statements Remove the TimeRange* Macro Variables

SAS Studio

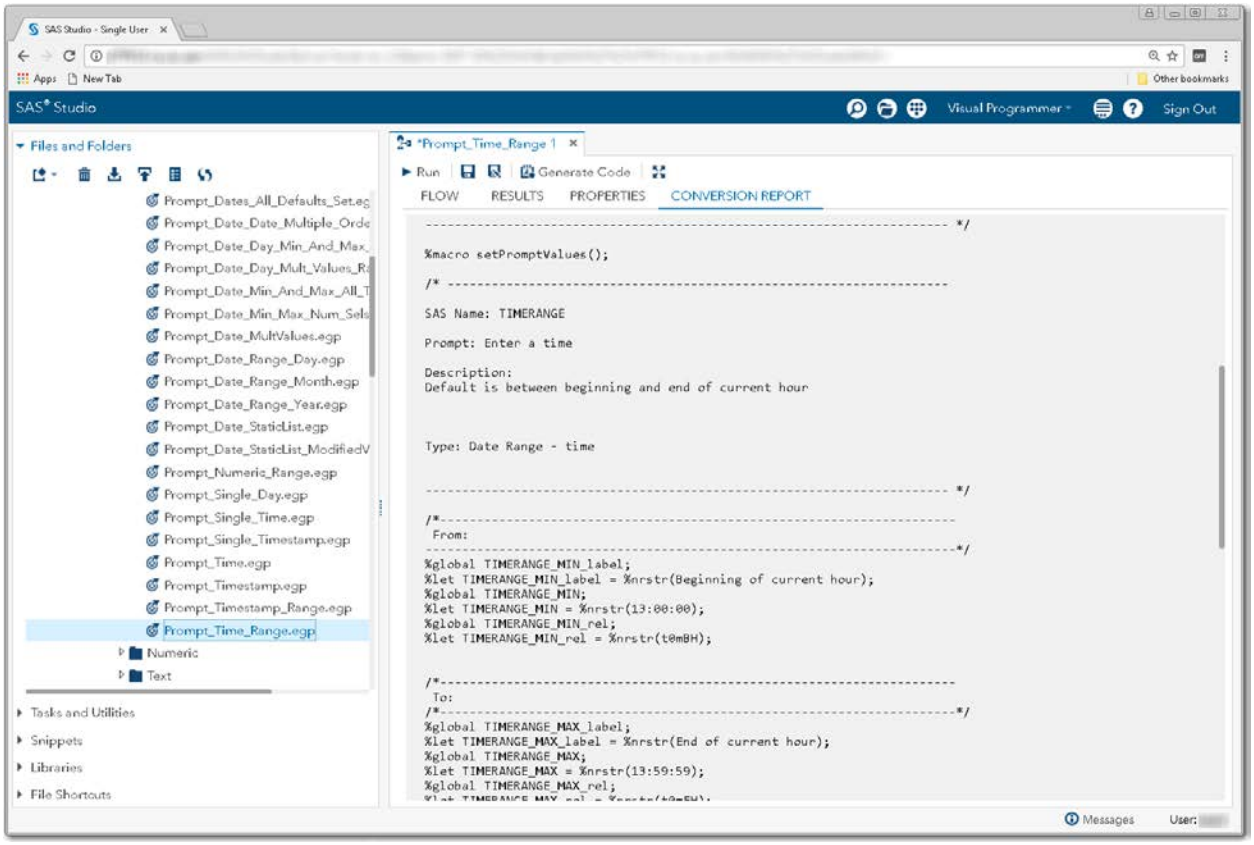
The following display shows code that is added to the converted Program node for the time range prompt in SAS Enterprise Guide.

These global time macro variables are created:

- TIMERANGE_MIN_rel
- TIMERANGE_MIN_label
- TIMERANGE_MIN
- TIMERANGE_MAX_rel
- TIMERANGE_MAX_label
- TIMERANGE_MAX

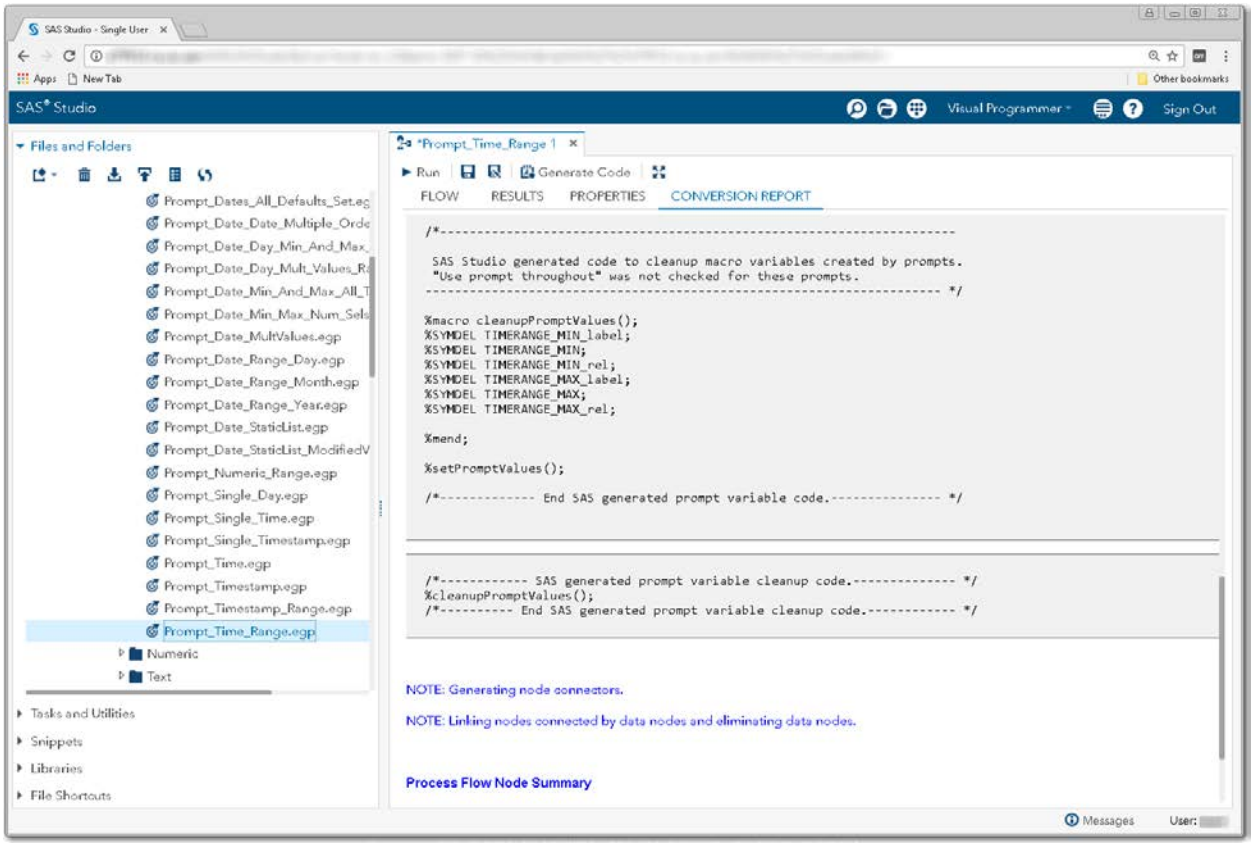
The %LET statements assign the default value to the TIMERANGE* macro variables.

If you want to run your process flow using different input for the TIMERANGE prompt, you must manually update the values of the macro variables in the %LET statements.



Display 200 – Code for Time Range Prompt

Because the **Use prompt value throughout project** option is not checked, the %SYMDEL statements remove the TIMERANGE macro variables.



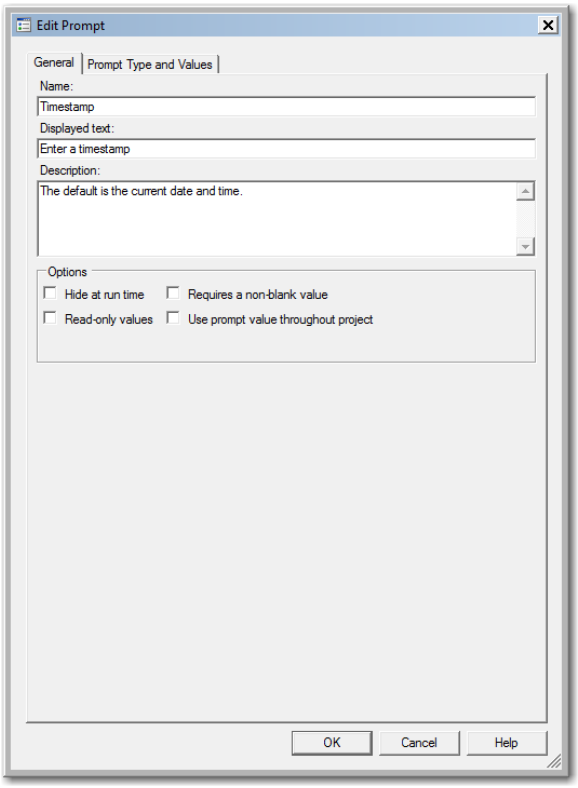
Display 201 - %SYMDEL Code Removes TIMERANGE* Macro Variables

Timestamp

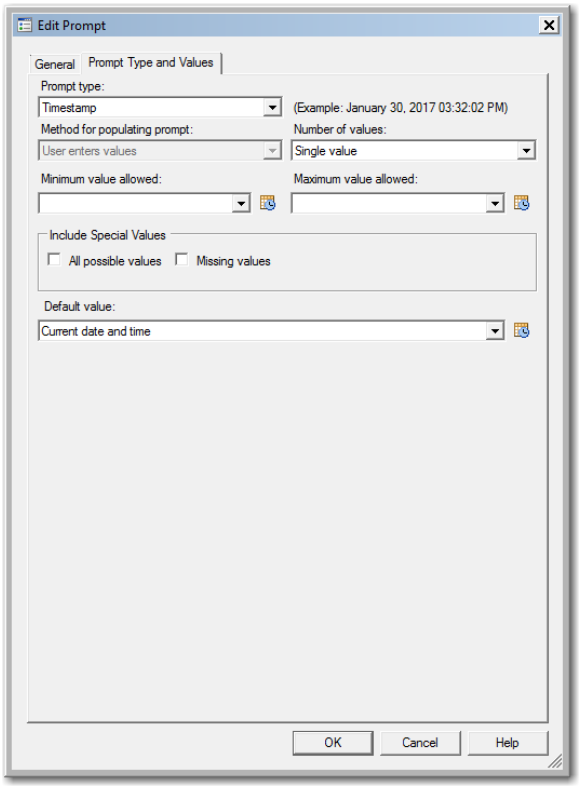
Single Timestamp

SAS Enterprise Guide

In this example, a timestamp prompt named Timestamp is defined as shown in the following two displays.

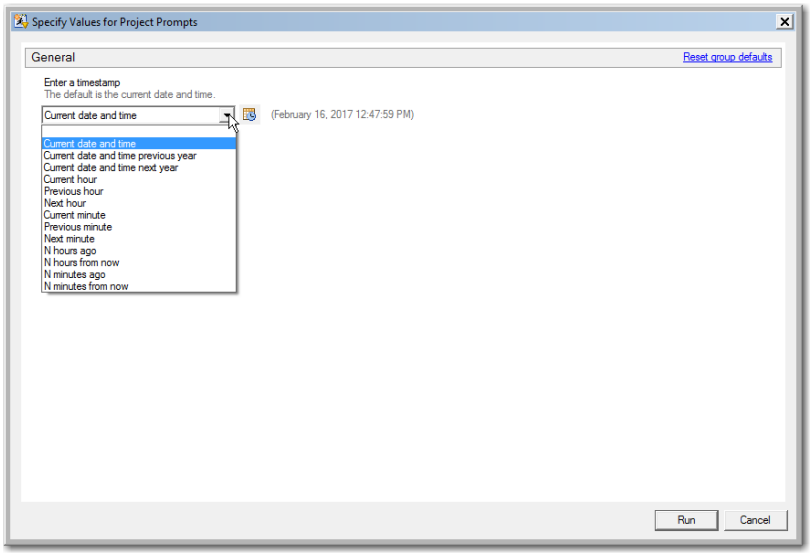


Display 202 - General Properties for Timestamp Prompt



Display 203 - Type and Values for Timestamp Prompt

When you run the Program node that depends on the prompt, the following dialog box appears:

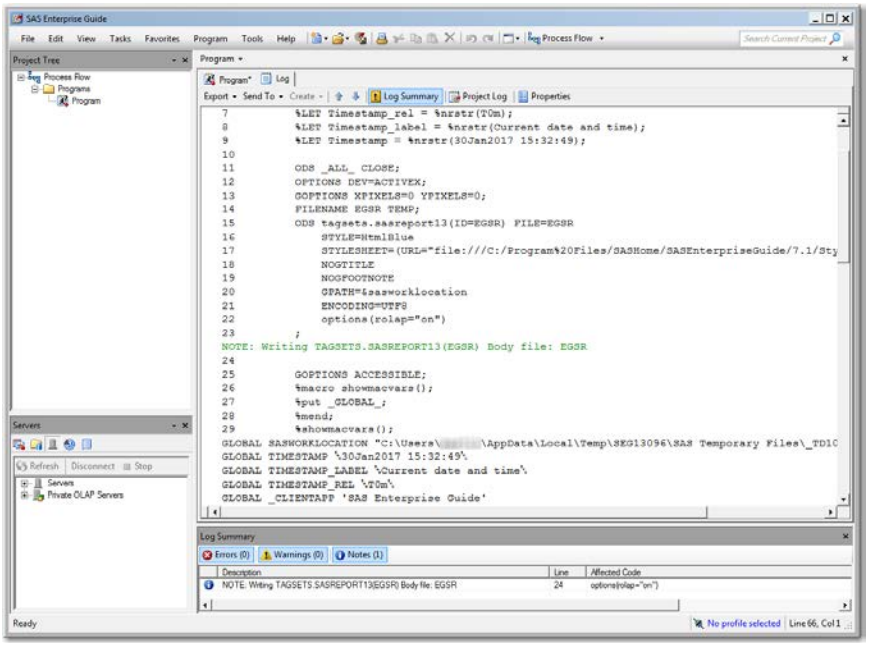


Display 204 - Timestamp Prompt in Prompt Dialog Box

If the user leaves the default value in the single timestamp prompt field, the following code is generated by SAS Enterprise Guide in the Program node that depends on the prompt.

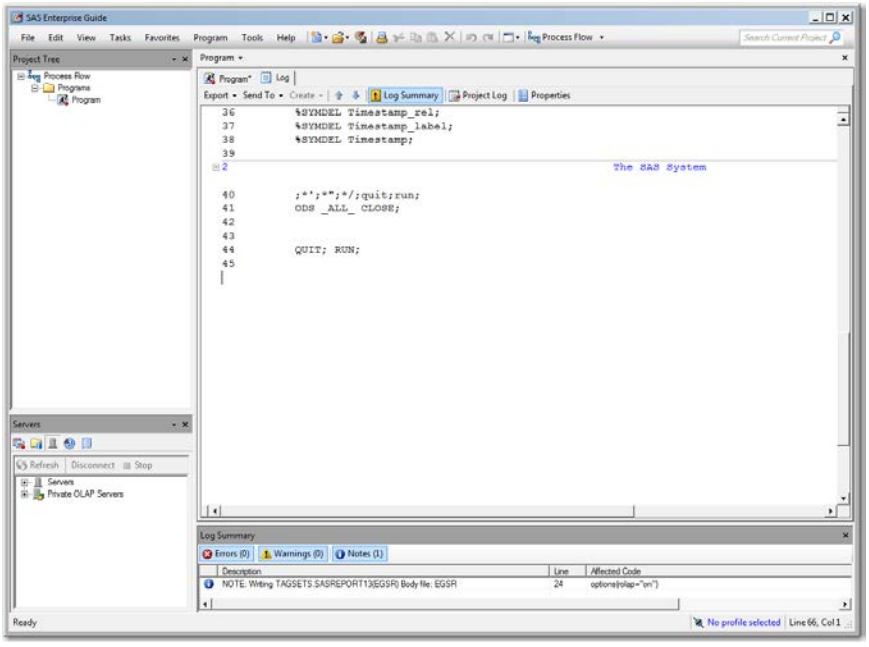
The %LET statements assign the value specified in the prompt dialog box to the Timestamp* macro variables.

The log of the Program node using the prompt definition displays the values of the global variables created by the prompt.



Display 205 - Global Macro Variables and %LET Statements for the Timestamp Prompt

Because the **Use prompt value throughout project** option is not checked for this prompt, the %SYMDEL statements remove the macro variables at the end of the program.



Display 206 - %SYMDEL Statements Remove Timestamp* Macro Variables

SAS Studio

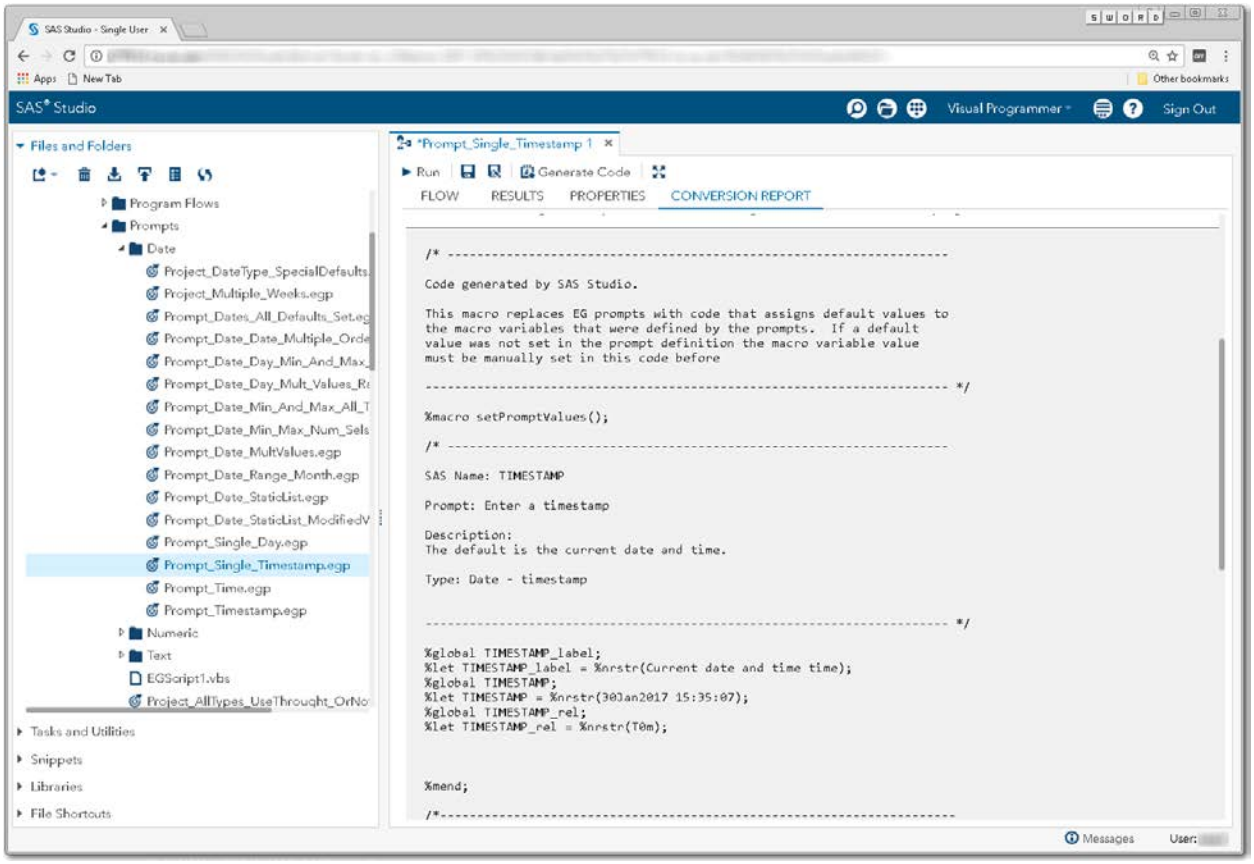
The following display shows the code that is added to the converted Program node for the time prompt in SAS Enterprise Guide.

These global timestamp macro variables are created:

- TIMESTAMP_rel
- TIMESTAMP_label
- TIMESTAMP

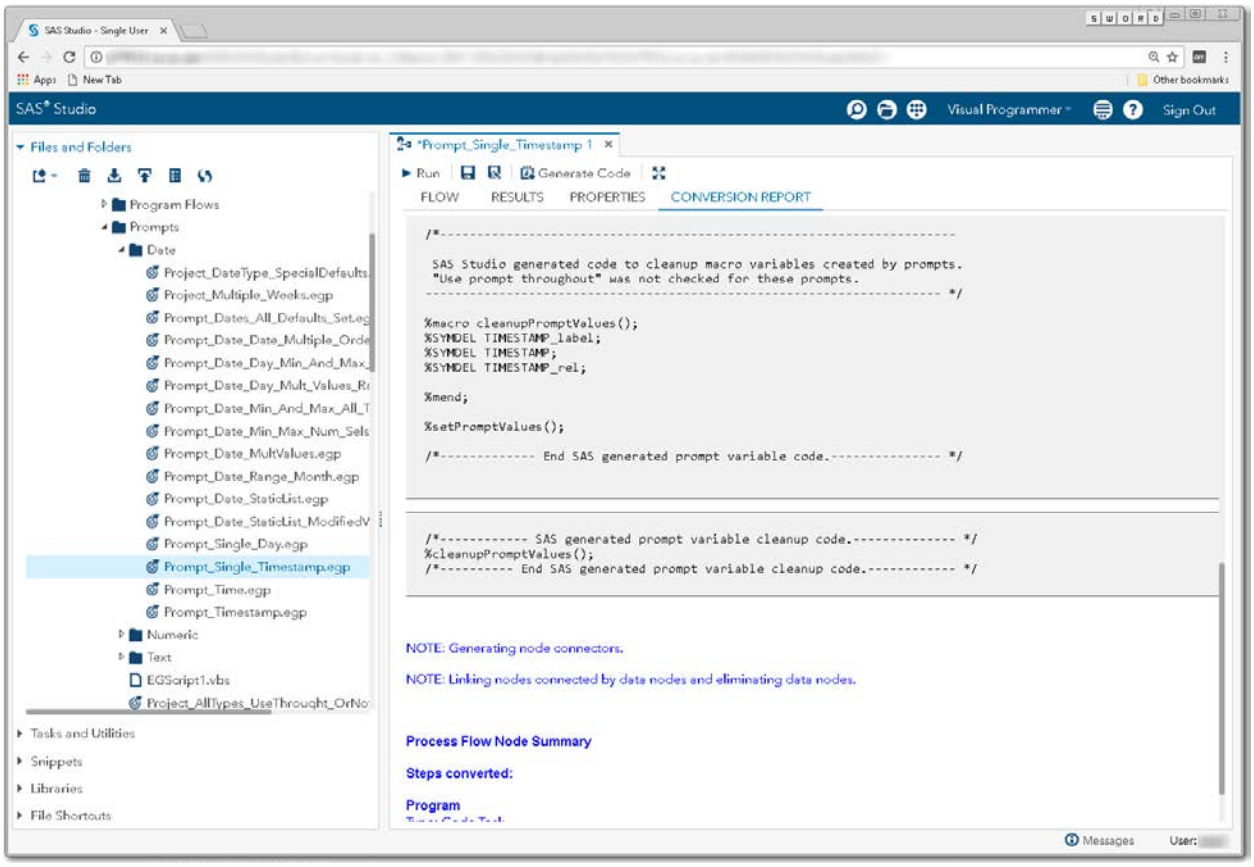
The %LET statements assign the default value to the TIMESTAMP* macro variables.

If you want to run your process flow using different values, you must manually update the values of the macro variables in the %LET statements.



Display 207 - Code for Timestamp Prompt

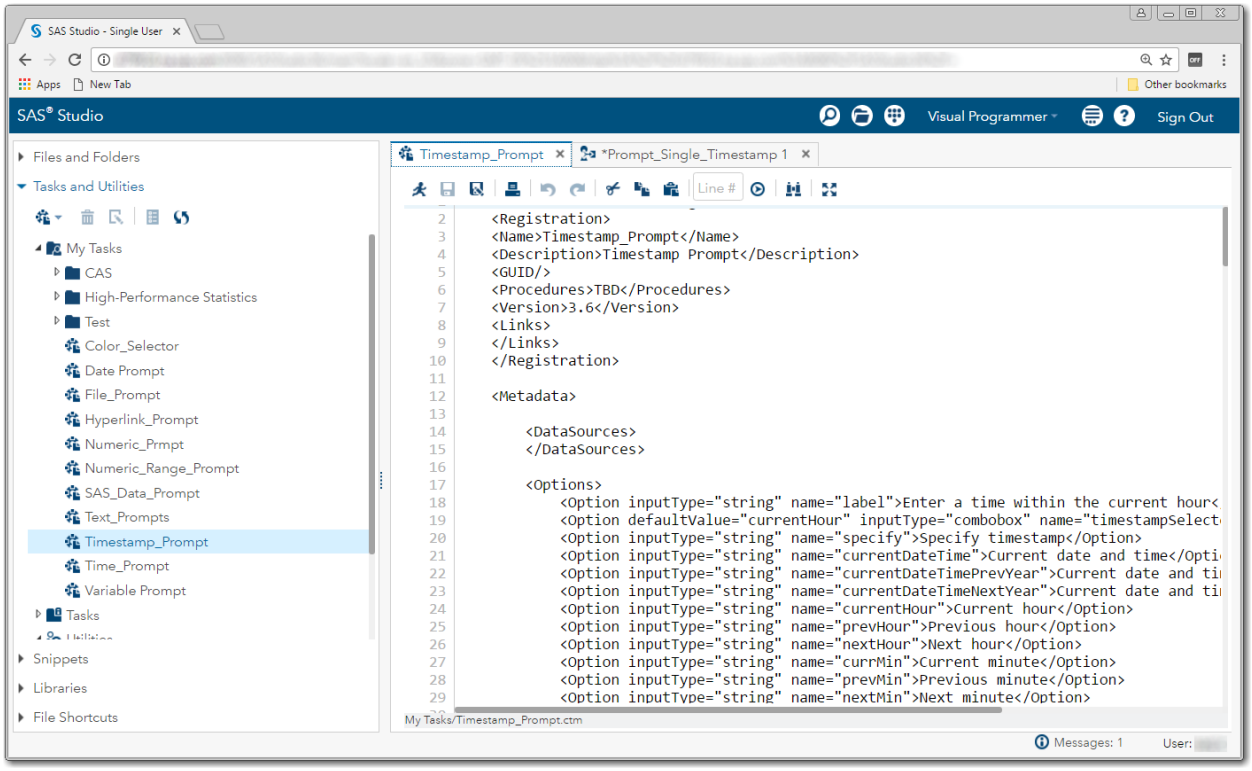
Because the **Use prompt value throughout project** option is not checked, the %SYMDEL statements remove the TIMESTAMP macro variables.



Display 208 - %SYMDEL Statements Remove TIMESTAMP* Macro Variables

Substituting a SAS Studio Task for Timestamp Prompt

1. Create a SAS Studio task with a control that represents the timestamp prompt.
   - Add controls as shown in the Timestamp_Prompt task.
   - Set the default value to the default value shown in the generated `setPromptValues()` macro in the converted Program node.
   - Change the strings for the input controls to match the strings specified in the prompt.



Display 209 - Replacement Task for Timestamp Prompt

The following code is an example of a task that could be used as the timestamp prompt.

```
<?xml version="1.0" encoding="UTF-8"?><Task runNLS="never" schemaVersion="5.1">
       <Registration>
       <Name>Timestamp_Prompt</Name>
       <Description>Timestamp Prompt</Description>
       <GUID/>
       <Procedures>TBD</Procedures>
       <Version>3.6</Version>
       <Links></Links>
       </Registration>

       <Metadata>

              <DataSources>
             </DataSources>

              <Options>

                   <Option inputType="string" name="label">
                        Enter a time within the current hour
                   </Option>
                  <Option defaultValue="currentHour" inputType="combobox"
                          name="timestampSelector">
                        Default is current time
                   </Option>
                   <Option inputType="string" name="specify">Specify timestamp</Option>
                   <Option inputType="string" name="currentDateTime">
                       Current date and time
                   </Option>
                   <Option inputType="string" name="currentDateTimePrevYear">
                       Current date and time previous year
                   <Option inputType="string" name="currentDateTimeNextYear">
                       Current date and time next year
                   </Option>
                   <Option inputType="string" name="currentHour">Current hour</Option>
                   <Option inputType="string" name="prevHour">Previous hour</Option>
                   <Option inputType="string" name="nextHour">Next hour</Option>
                   <Option inputType="string" name="currMin">Current minute</Option>
                   <Option inputType="string" name="prevMin">Previous minute</Option>
                   <Option inputType="string" name="nextMin">Next minute</Option>
                   <Option inputType="string" name="nHoursAgo">N hours ago</Option>
                   <Option inputType="string" name="nHoursFromNow">
                       N hours from now
                   </Option>
                   <Option inputType="string" name="nMinsAgo">N minutes ago</Option>
                   <Option inputType="string" name="nMinsFromNow">
                       N minutes from now
                   </Option>
                   <Option inputType="inputtext" name="timeInput"
                           promptMessage="Example: 14Feb2017 01:23:45 PM" />
                   <Option defaultValue="N" inputType="numbertext"
                           invalidMessage="Invalid value. Enter a positive integer."
                           minValue="0"
                           missingMessage="Enter number of minutes"
                           name="numMinutes"
                           promptMessage="Enter number of minutes"
                           rangeMessage=
                              "This number is out of range.  Enter a positive number.">
                          Number of minutes:
                    </Option>
```

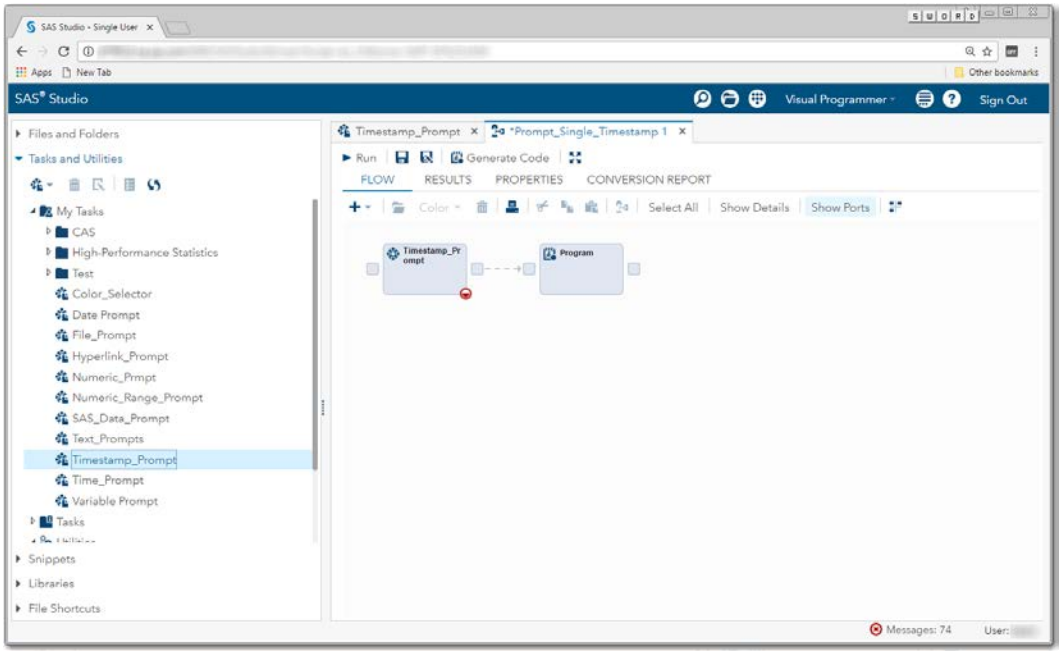
```
            <Option defaultValue="N" inputType="numbertext"
                   invalidMessage="Invalid value. Enter a positive integer."
                   minValue="0" missingMessage="Enter number of hours"
                   name="numHours"
                   promptMessage="Enter number of hours"
                   rangeMessage=
                      "This number is out of range.  Enter a positive number.">
                   Number of hours:
              </Option>

        </Options>

  </Metadata>


  <UI>

      <OptionItem option="label"/>
        <OptionChoice option="timestampSelector">
              <OptionItem option="specify"/>
              <OptionItem option="currentDateTime"/>
              <OptionItem option="currentDateTimePrevYear"/>
              <OptionItem option="currentDateTimeNextYear"/>
              <OptionItem option="currentHour"/>
              <OptionItem option="prevHour"/>
              <OptionItem option="nextHour"/>
              <OptionItem option="currMin"/>
              <OptionItem option="prevMin"/>
              <OptionItem option="nextMin"/>
              <OptionItem option="nHoursAgo"/>
              <OptionItem option="nHoursFromNow"/>
              <OptionItem option="nMinsAgo"/>
              <OptionItem option="nMinsFromNow"/>
        </OptionChoice>

        <OptionItem option="timeInput"/>
        <OptionItem option="numMinutes"/>
        <OptionItem option="numHours"/>

  </UI>

<Dependencies>

    <Dependency condition="($timestampSelector == 'specify')">
         <Target action="hide" conditionResult="false" option="timeInput"/>
         <Target action="show" conditionResult="true" option="timeInput"/>
    </Dependency>
    <Dependency condition="(($timestampSelector == 'nMinsAgo') ||
                           ($timestampSelector == 'nMinsFromNow'))">
         <Target action="hide" conditionResult="false" option="numMinutes"/>
         <Target action="show" conditionResult="true" option="numMinutes"/>
    </Dependency>
    <Dependency condition="(($timestampSelector == 'nHoursAgo') ||
                           ($timestampSelector == 'nHoursFromNow'))">
         <Target action="hide" conditionResult="false" option="numHours"/>
         <Target action="show" conditionResult="true" option="numHours"/>
    </Dependency>

 </Dependencies>
```

```
    <CodeTemplate>
        <![CDATA[

%global timestamp;
%global timestamp_rel;
%global timestamp_label;

#if ($timestampSelector == 'specify')
   %let timestamp = $timeInput;
   %let timestamp_label = $timeInput;  /* Format this as you see fit */
   %symdel timestamp_rel;
#else

#if ($timestampSelector == 'currentDateTime')
   %let timestamp = %sysfunc(datetime(), DATETIME20.);
   %let timestamp = %sysfunc(catx(%str( ), %substr(&timestamp,1,9),
                             %substr(&timestamp,11,8)));
   %let timestamp_label = Current date and time;
   %let timestamp_rel = T0m;
#else

#if ($timestampSelector == 'currentDateTimePrevYear')
   %global year;
   %global datetime;
   %let datetime = %sysfunc(datetime());
   %let datetime = %sysfunc(intnx(dtyear, &datetime, -1), DATETIME20.);
   %let year = %substr(&datetime, 6,4);

   %let datetime = %sysfunc(datetime(), DATETIME20.);
   %let timestamp = %sysfunc(catx(%str(), %substr(&datetime, 1, 5), &year));
   %let timestamp = %sysfunc(catx(%str( ), &timestamp, %substr(&datetime, 11, 8)));
   %symdel year;
   %symdel datetime;

   %let timestamp_label = Current date and time previous year;
   %let timestamp_rel = T-1Y;
#else

#if ($timestampSelector == 'currentDateTimeNextYear')
   %global year;
   %global datetime;
   %let datetime = %sysfunc(datetime());
   %let datetime = %sysfunc(intnx(dtyear, &datetime, 1), DATETIME20.);
   %let year = %substr(&datetime, 6,4);

   %let datetime = %sysfunc(datetime(), DATETIME20.);
   %let timestamp = %sysfunc(catx(%str(), %substr(&datetime, 1, 5), &year));
   %let timestamp = %sysfunc(catx(%str( ), &timestamp, %substr(&datetime, 11, 8)));
   %symdel year;
   %symdel datetime;
   %let time_label = Current date and time next year;
   %let timestamp_rel = T1Y;
#else

#if ($timestampSelector == 'currentHour')
   %let timestamp = %sysfunc(datetime());
   %let timestamp = %sysfunc(intnx(hour, &timestamp, 0, b), DATETIME20.);
   %let timestamp = %sysfunc(catx(%str( ), %substr(&timestamp,1,9),
                             %substr(&timestamp,11,8)));
   %let timestamp_label = Current hour;
   %let timestamp_rel = H0H;
#else
```

```
#if ($timestampSelector == 'currMin')
   %let timestamp = %sysfunc(datetime());
   %let timestamp = %sysfunc(intnx(minute, &timestamp, 0, b), DATETIME20.);
   %let timestamp = %sysfunc(catx(%str( ), %substr(&timestamp,1,9),
                            %substr(&timestamp,11,8)));
   %let timestamp_label = Current minute;
   %let timestamp_rel = mom;
#else

#if ($timestampSelector == 'prevHour')
   %let timestamp = %sysfunc(datetime());
   %let timestamp = %sysfunc(intnx(dthour, &timestamp, -1), DATETIME20.);
   %let timestamp = %sysfunc(catx(%str( ), %substr(&timestamp,1,9),
                            %substr(&timestamp,11,8)));
   %let timestamp_label = Previous hour;
   %let timestamp_rel = H-1H;
#end

#if ($timestampSelector == 'nextHour')
   %let timestamp = %sysfunc(datetime());
   %let timestamp = %sysfunc(intnx(dthour, &timestamp, 1), DATETIME20.);
   %let timestamp = %sysfunc(catx(%str( ), %substr(&timestamp,1,9),
                            %substr(&timestamp,11,8)));
   %let timestamp_label = Next hour;
   %let timestamp_rel = H1H;
#end

#if ($timestampSelector == 'prevMin')
   %let timestamp = %sysfunc(datetime());
   %let timestamp = %sysfunc(intnx(dtminute, &timestamp, -1), DATETIME20.);
   %let timestamp = %sysfunc(catx(%str( ), %substr(&timestamp,1,9),
                            %substr(&timestamp,11,8)));
   %let timestamp_label = Previous minute;
   %let timestamp_rel = m-1m;
#end

#if ($timestampSelector == 'nextMin')
   %let timestamp = %sysfunc(datetime());
   %let timestamp = %sysfunc(intnx(dtminute, &timestamp, 1), DATETIME20.);
   %let timestamp = %sysfunc(catx(%str( ), %substr(&timestamp,1,9),
                            %substr(&timestamp,11,8)));
   %let timestamp_label = Next minute;
   %let timestamp_rel = m1m;
#end

#if ($timestampSelector == 'nHoursAgo')
   %let timestamp = %sysfunc(datetime());
   %let timestamp = %sysfunc(intnx(dtsecond, &timestamp, -${numHours}*3600),
                               DATETIME20.);
   %let timestamp = %sysfunc(catx(%str( ), %substr(&timestamp,1,9),
                              %substr(&timestamp,11,8)));
   %let timestamp_label = $numHours hours ago;
   %let timestamp_rel = T-${numHours}H;
#end
```

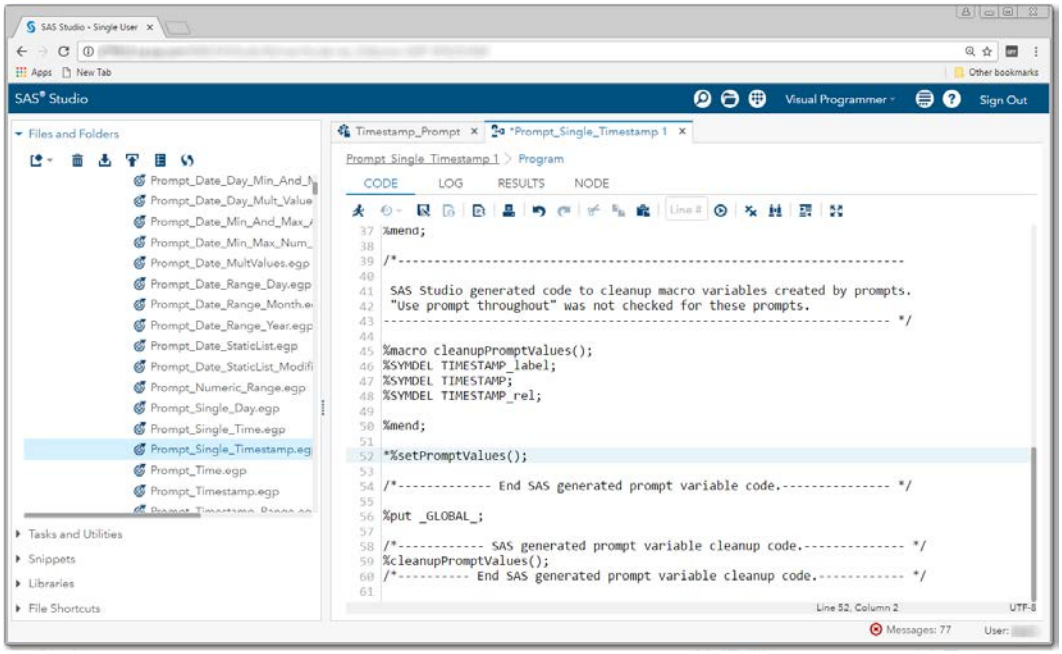
```
#if ($timestampSelector == 'nHoursFromNow')
   %let timestamp = %sysfunc(datetime());
   %let timestamp = %sysfunc(intnx(dtsecond, &timestamp, ${numHours}*3600),
                             DATETIME20.);
   %let timestamp = %sysfunc(catx(%str( ), %substr(&timestamp,1,9),
                             %substr(&timestamp,11,8)));
   %let timestamp_label = $numHours hours from now;
   %let timestamp_rel = T${numHours}H;
#end

#if ($timestampSelector == 'nMinsAgo')
   %let timestamp = %sysfunc(datetime());
   %let timestamp = %sysfunc(intnx(dtsecond, &timestamp, -${numMinutes}*60),
                                  DATETIME20.);
   %let timestamp = %sysfunc(catx(%str( ), %substr(&timestamp,1,9),
                             %substr(&timestamp,11,8)));
   %let timestamp_label = $numMinutes minutes ago;
   %let timestamp_rel = T-${numMinutes}m;
#end

#if ($timestampSelector == 'nMinsFromNow')
   %let timestamp = %sysfunc(datetime());
   %let timestamp = %sysfunc(intnx(dtsecond, &timestamp, ${numMinutes}*60),
                             DATETIME20.);
   %let timestamp = %sysfunc(catx(%str( ), %substr(&timestamp,1,9),
                             %substr(&timestamp,11,8)));
   %let timestamp_label = $numMinutes minutes from now;
   %let timestamp_rel = T${numMinutes}m;
#end

#end
#end
#end
#end
#end
#end
        ]]>
    </CodeTemplate>
</Task>
```

2. Save the prompt replacement task to your **My Tasks** folder.
3. Drag the task from **My Tasks** into your converted process flow.
4. Link the output port of the task to the input port of the converted Program node.



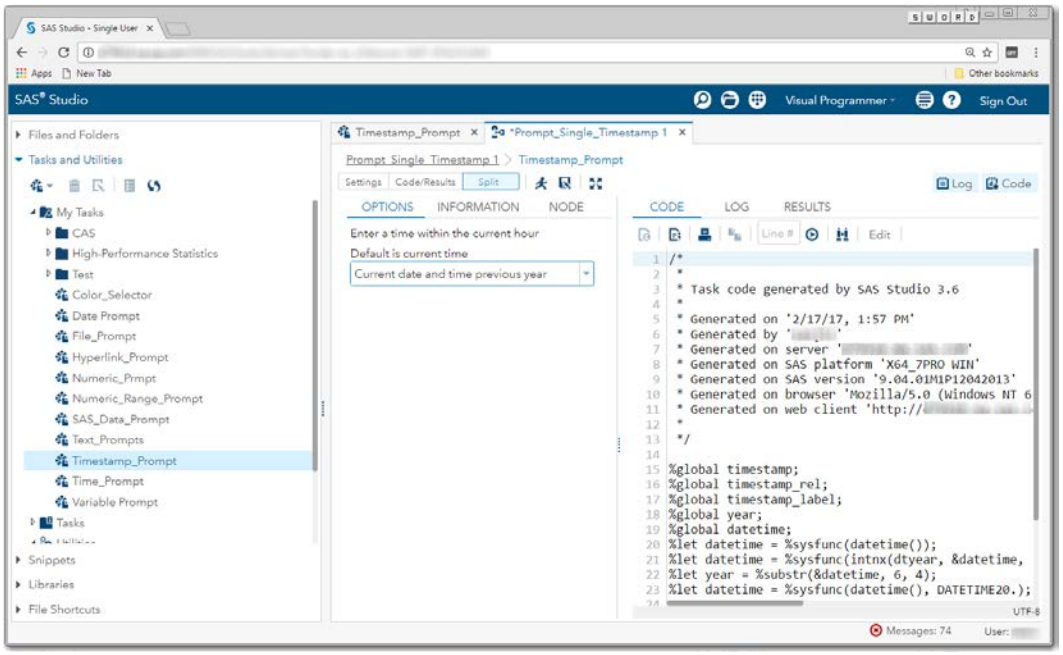
Display 210 – Timestamp_Prompt Task Linked to Program Node

5. Comment out the `%setPromptValues()` macro call from the converted Program node. The macro code generated by the Timestamp_Prompt task replaces this code.



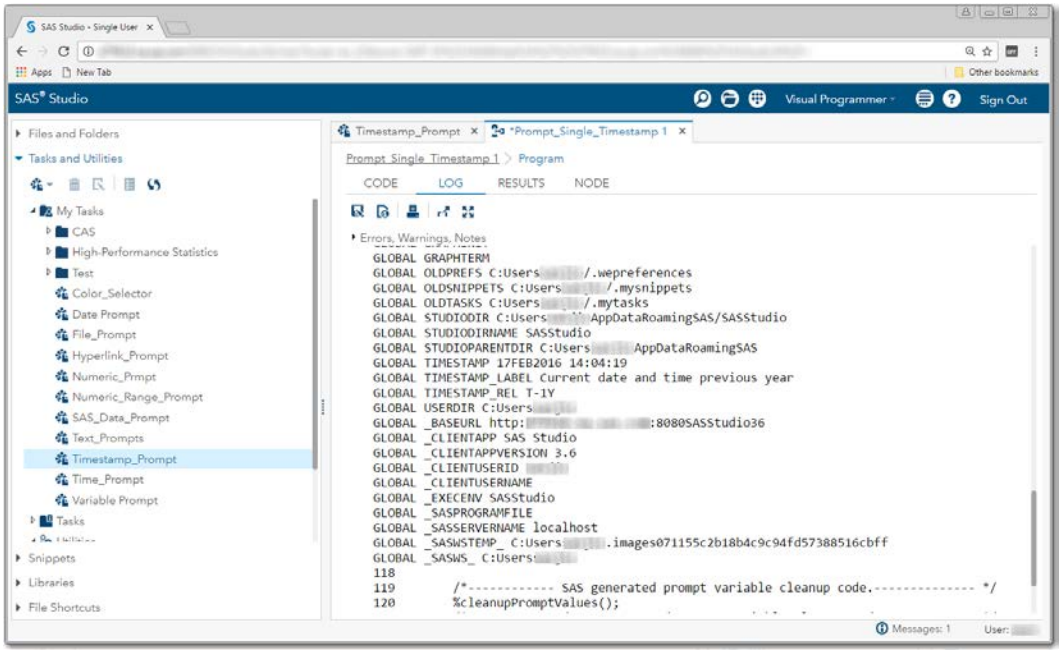
Display 211 – Commented out %setPromptValues in Program Node

To run your flow with a different timestamp value than the default value, open the Timestamp_Prompt node and specify a different timestamp.



Display 212 - Timestamp Prompt Task

When you run the process flow, the global Timestamp* variables are set to the values specified in the task.

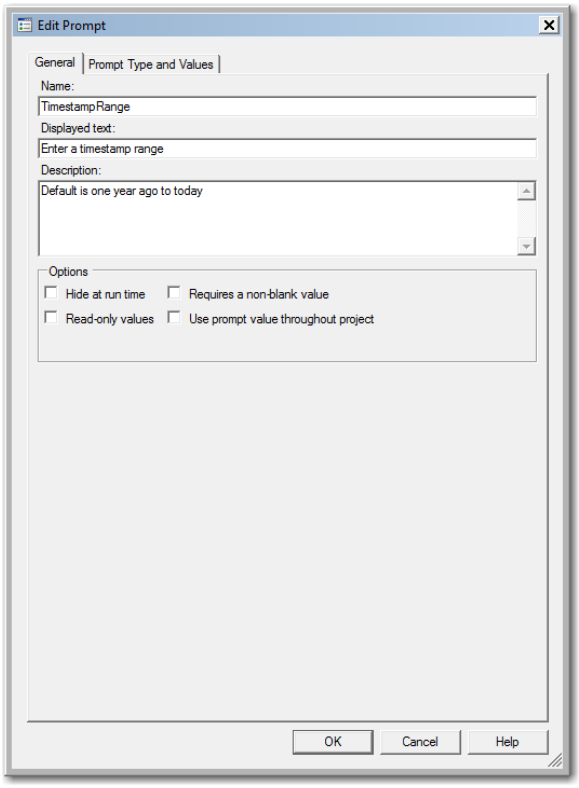


Display 213 – Timestamp Prompt Variables with Updated Values

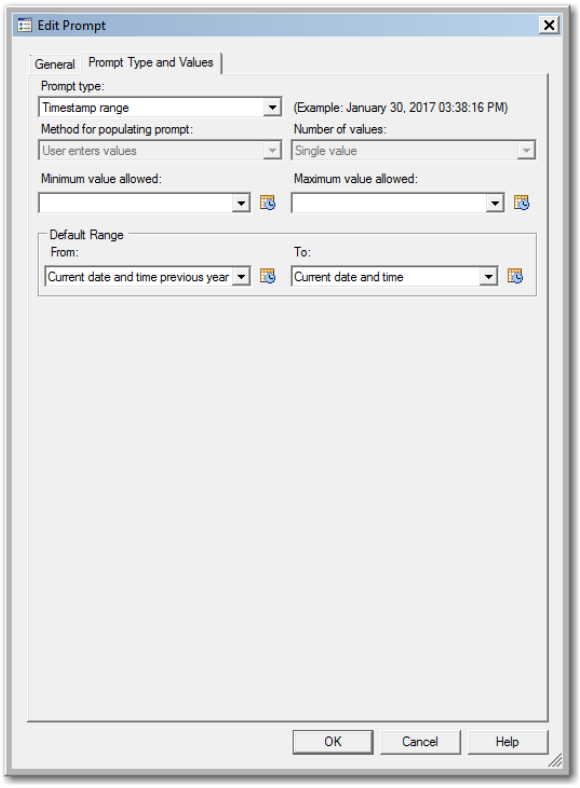
Timestamp Range

SAS Enterprise Guide

In this example, a timestamp range prompt named TimestampRange is defined as shown in the following two displays.

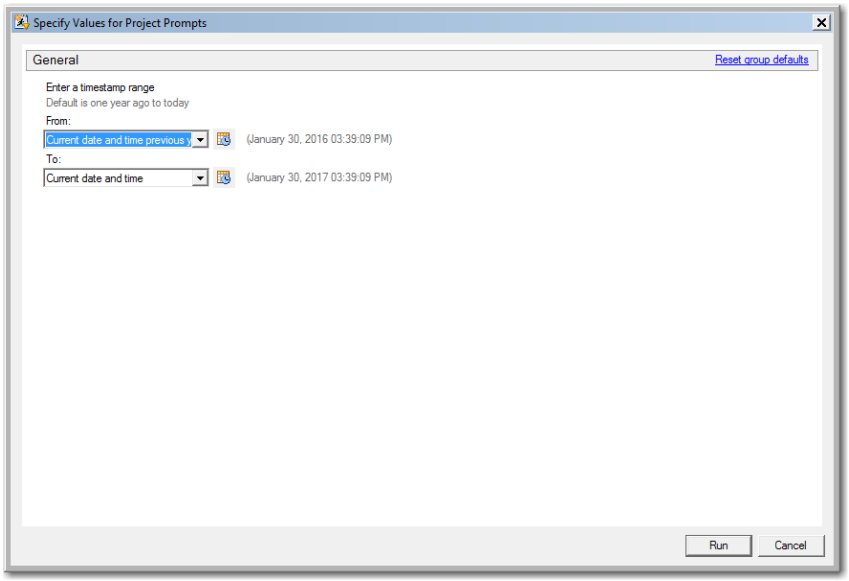


Display 214 - General Properties for Timestamp Range Prompt



Display 215 - Type and Values for Timestamp Range Prompt

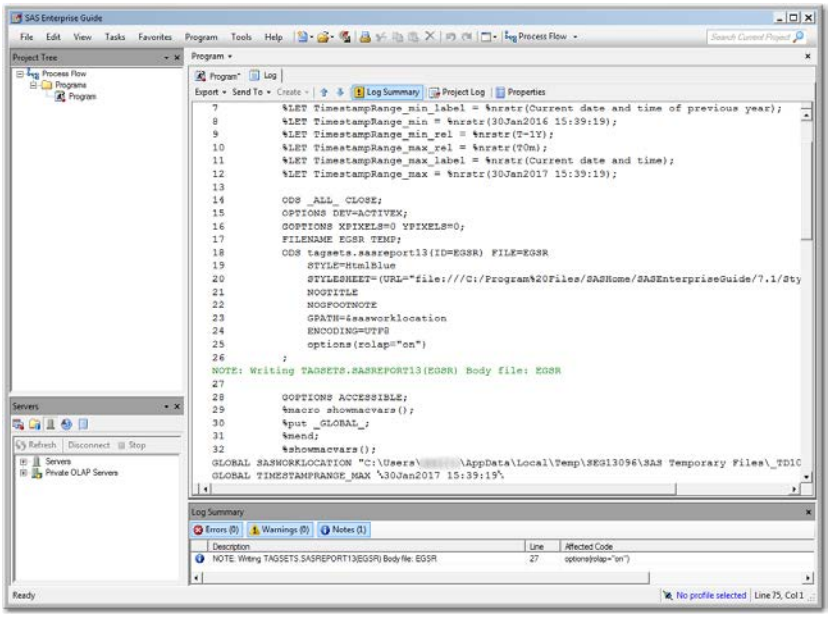
When you run the Program node that depends on this prompt, the following dialog box appears:



Display 216 - Timestamp Range Prompt in Prompt Dialog Box

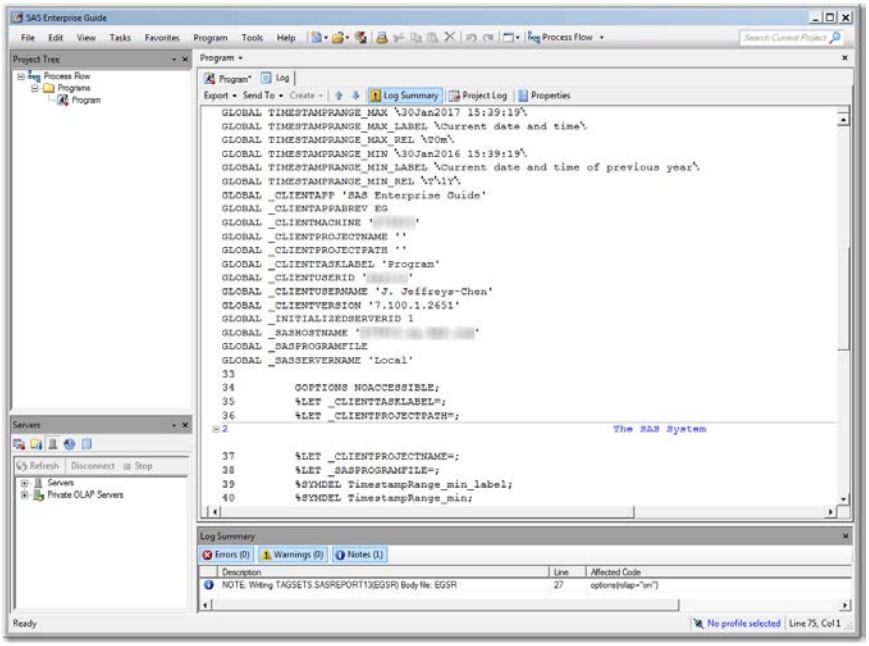
If the user leaves the default value in the timestamp range prompt field, the following code is generated by SAS Enterprise Guide in the Program node that depends on the prompt.

The %LET statements assign the values specified in the prompt dialog box to the TimestampRange* macro variables.



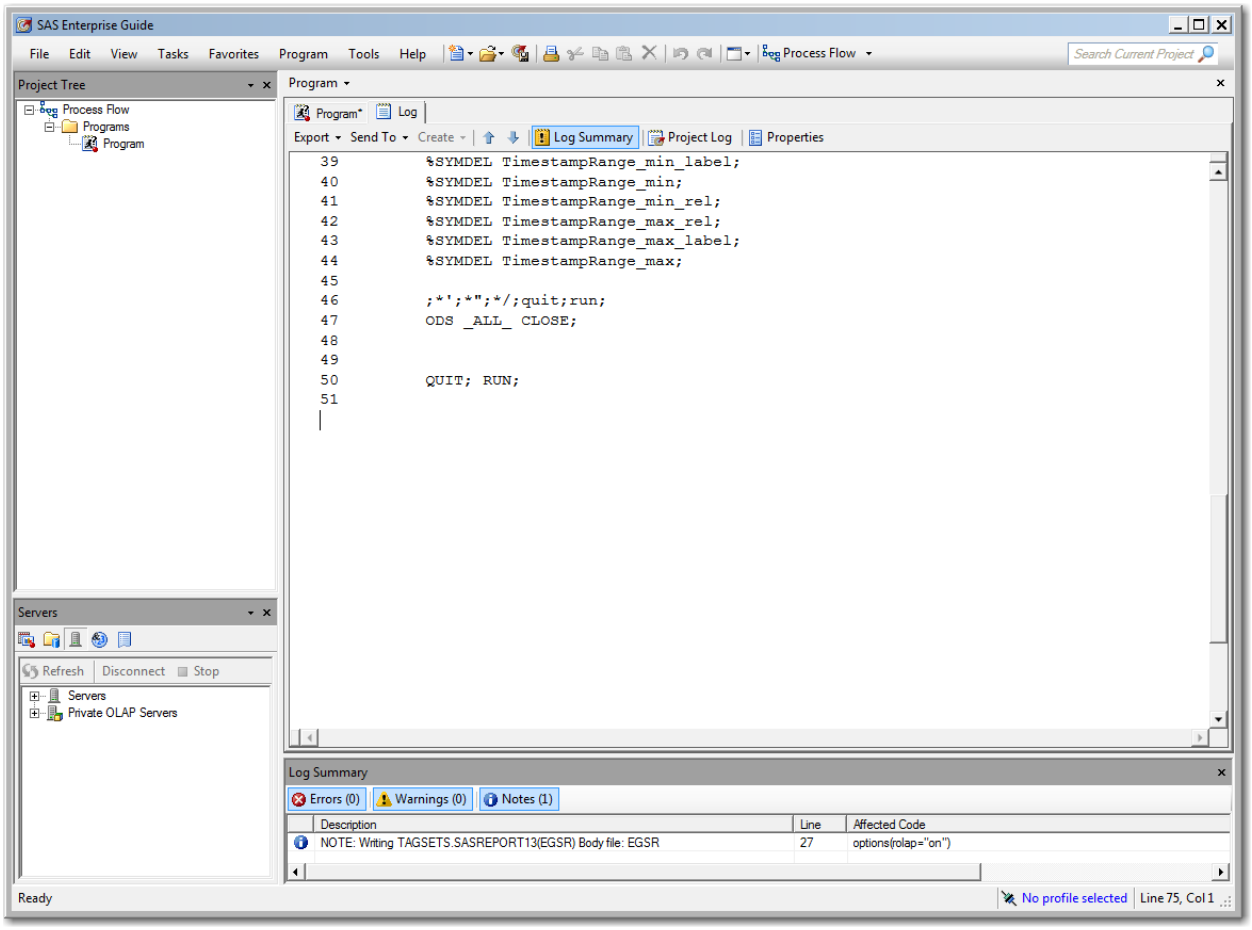
Display 217 - %LET Statements for Timestamp Range Prompt

The log of the Program node using the prompt definition displays the values of the global variables created by the prompt.



Display 218 - Global Macro Variables for Timestamp Range Prompt

Because the **Use prompt value throughout project** option is not checked for this prompt, the %SYMDEL statements remove the macro variables at the end of the program.



Display 219 - %SYMDEL Statements Remove TimestampRange* Macro Variables

SAS Studio

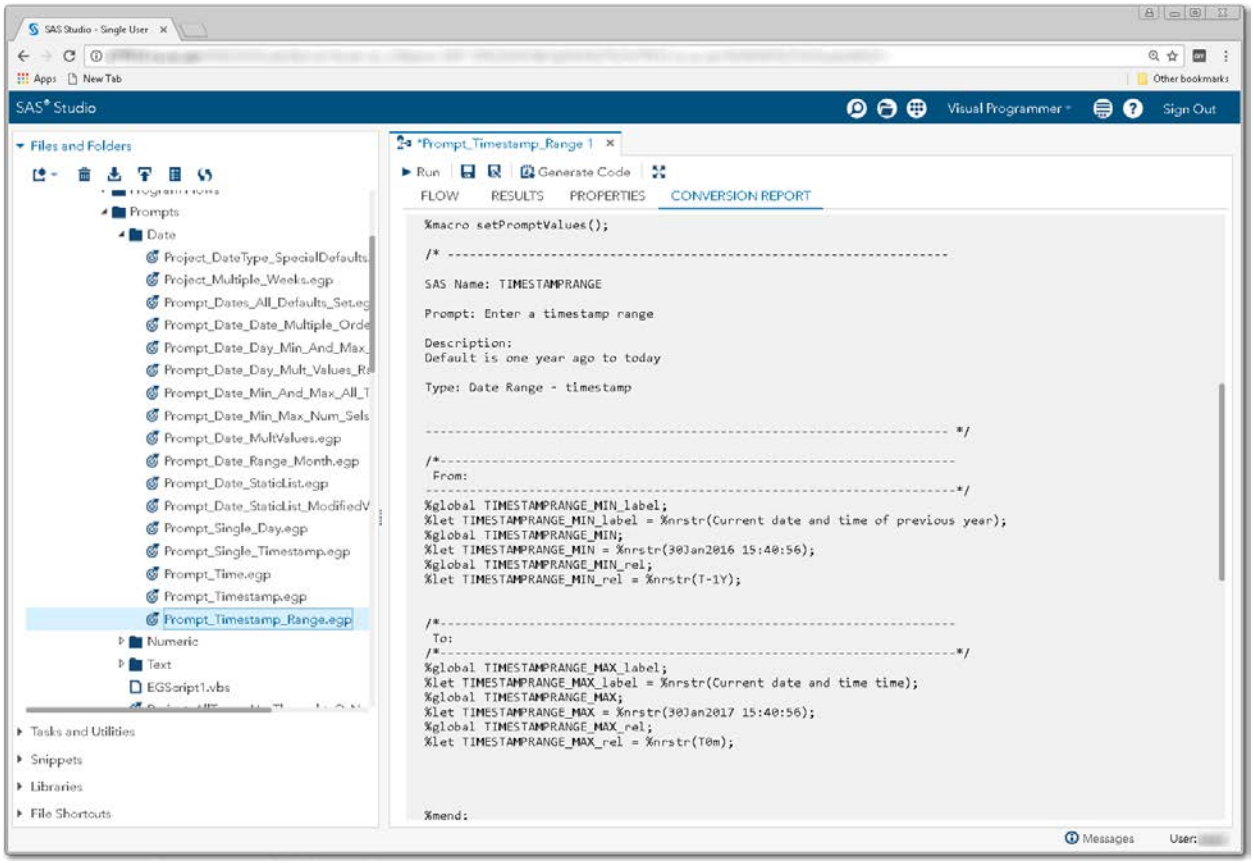
The following display shows the code that is added to the converted Program node for the timestamp range prompt in SAS Enterprise Guide.

These global macro variables are created:

- TIMESTAMPRANGE_MIN_rel
- TIMESTAMPRANGE_MIN_label
- TIMESTAMPRANGE_MIN
- TIMESTAMPRANGE_MAX_rel
- TIMESTAMPRANGE_MAX_label
- TIMESTAMPRANGE_MAX

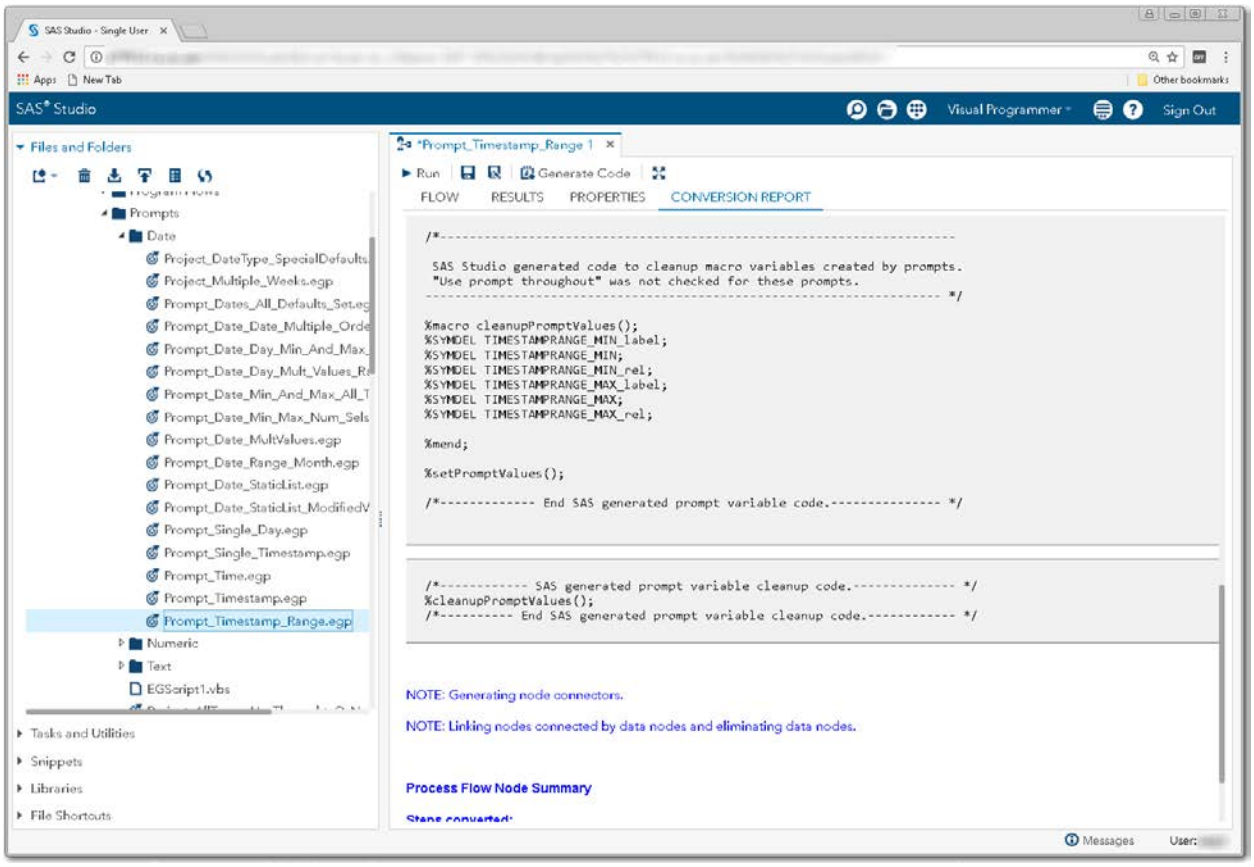
The %LET statements assign the default value to the TIMESTAMPRANGE* macro variables.

If you want to run your process flow using different values for the TIMESTAMPRANGE prompt, you must manually update the values of the macro variables in the %LET statements.



Display 220 - Macro Code for Timestamp Range Prompt

Because the **Use prompt value throughout project** is not checked, the %SYMDEL statements remove the TIMESTAMPRANGE macro variables.



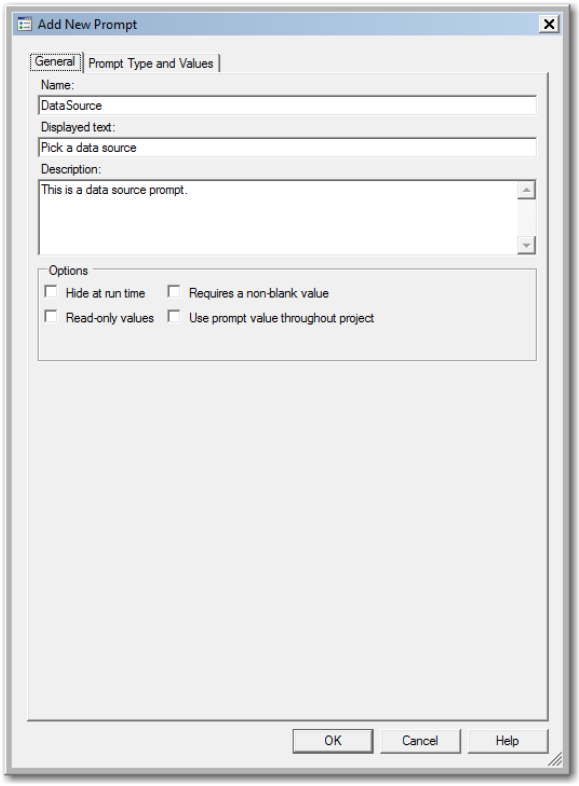
Display 221 - %SYMDEL Statements Remove the TIMESTAMPRANGE* Macro Variables

Data

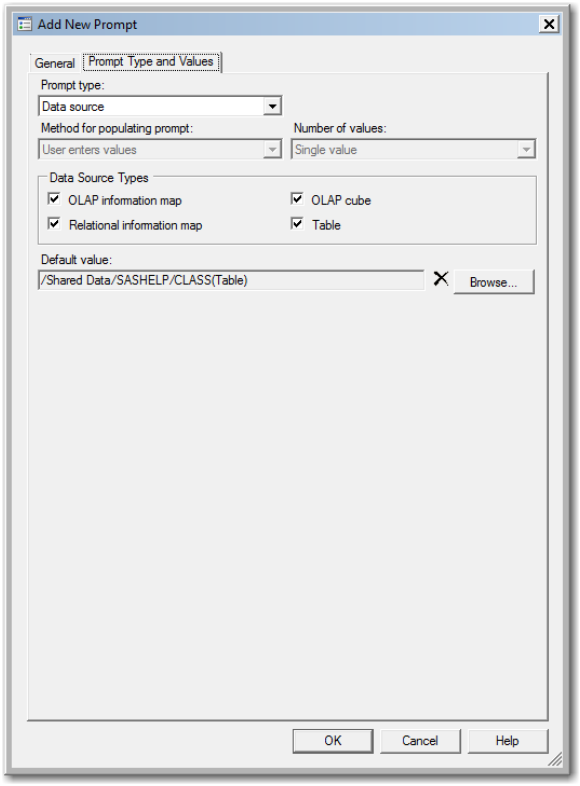
Data Source

SAS Enterprise Guide

In this example, a data source prompt named DataSource is defined as shown in the following two displays.

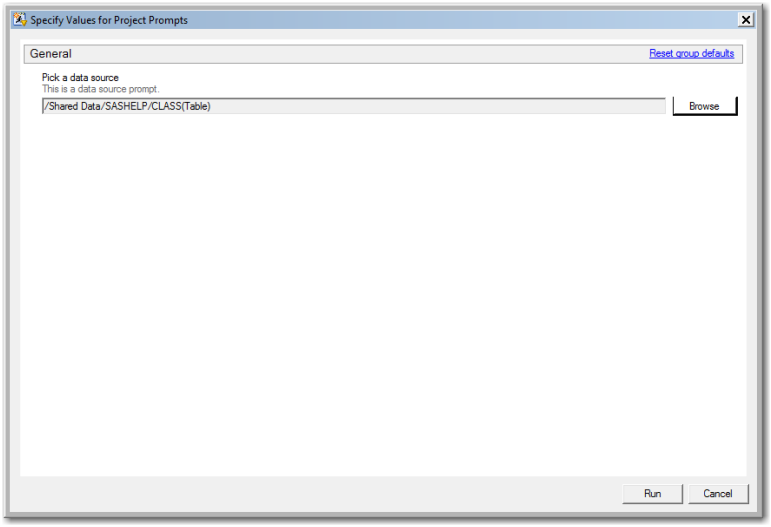


Display 222 - General Properties for Data Source Prompt



Display 223 - Type and Values for Data Source Prompt

When you run the Program node that depends on the prompt, the following dialog box appears:

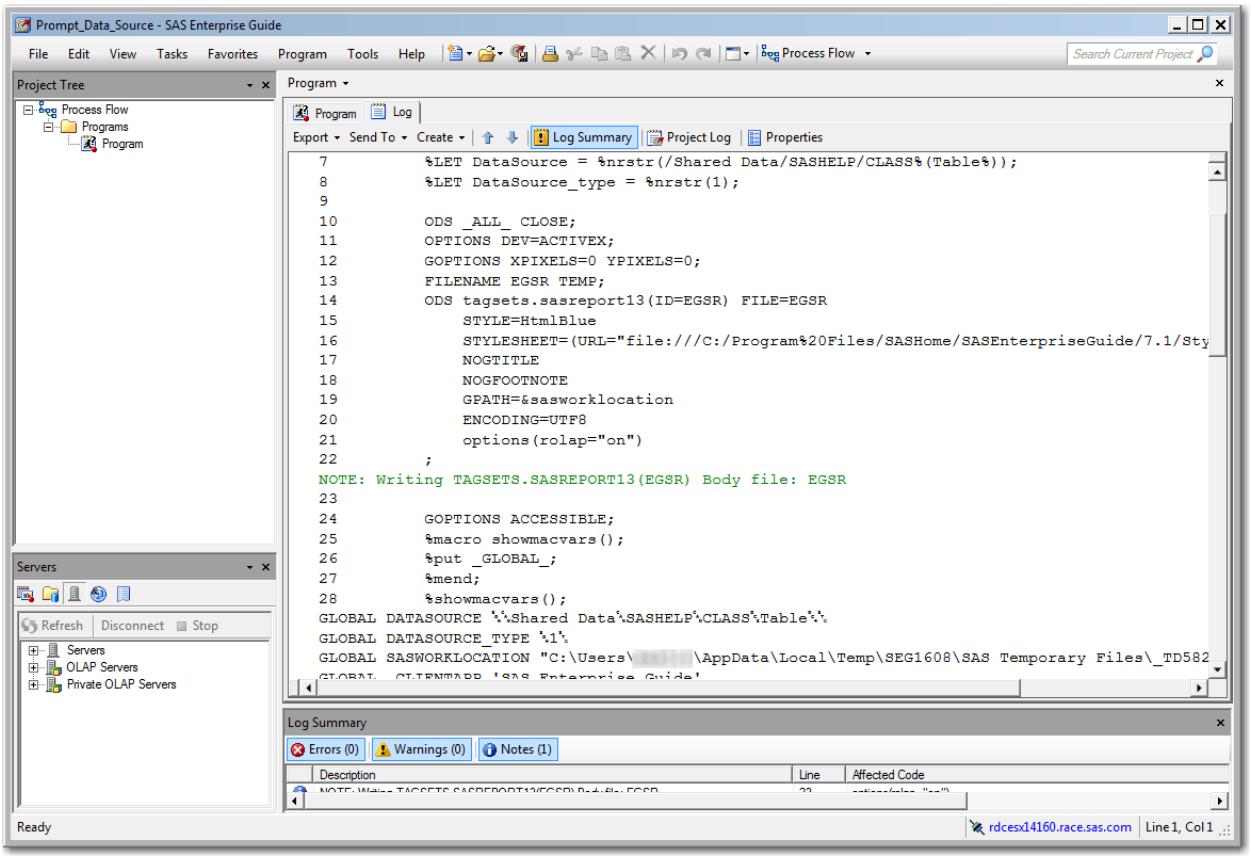


Display 224 - Data Source Prompt in Prompt Dialog Box

If the user leaves the default value in the data source value prompt field, the following code is generated by SAS Enterprise Guide in the Program node that depends on the prompt.

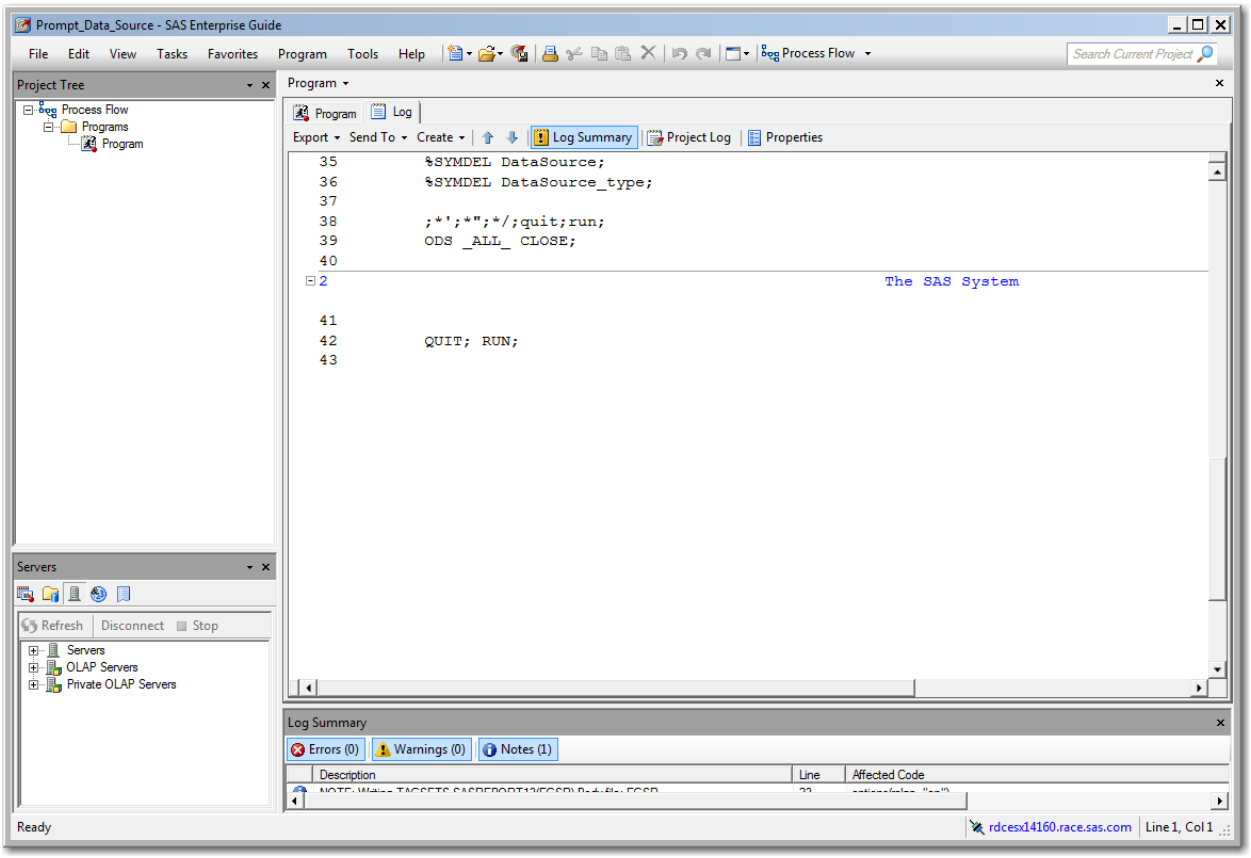
The %LET statements assign the values specified in the prompt dialog box to the DataSource* macro variables.

The log of the Program node using the prompt definition displays the values of the global variables created by the prompt.



Display 225 - %LET Statements for Data Source Prompt

Because the **Use prompt value throughout project** option is not checked for this prompt, the %SYMDEL statements remove the macro variables at the end of the program.



Display 226 - %SYMDEL Statements Remove DataSource* Macro Variables

SAS Studio

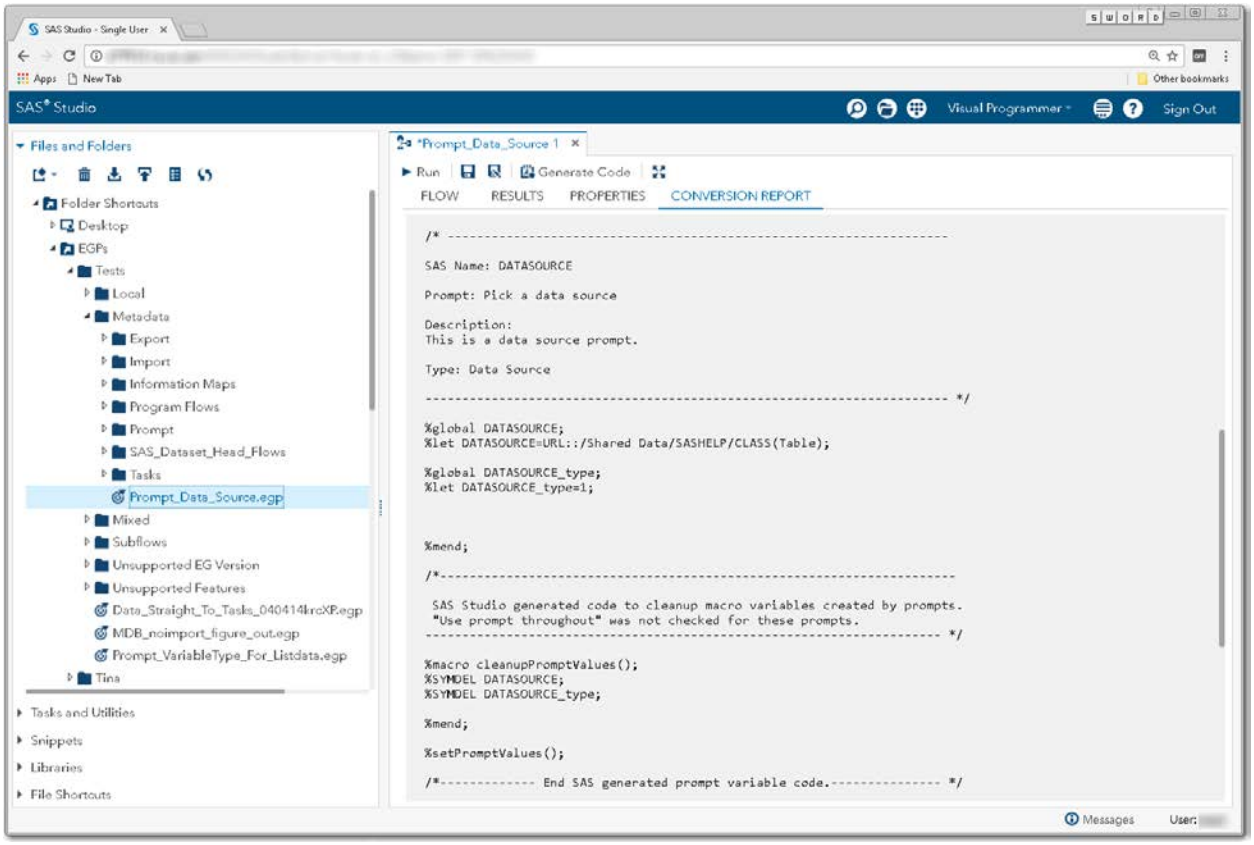
The following display shows the code that is added to the converted Program node for data source prompt in SAS Enterprise Guide.

These global variables are created:

- DATASOURCE
- DATASOURCE_type

The %LET statements assign the default value to the DATASOURCE* macro variables.

If you want to run your process flow using different values for the DATASOURCE prompt, you must manually update values of the macro variables in the %LET statements.



Display 227 - Macro Variable Code for Data Source Prompt

Note: Currently, the _type macro variable is always set to 1. This value might not be correct for all cases.

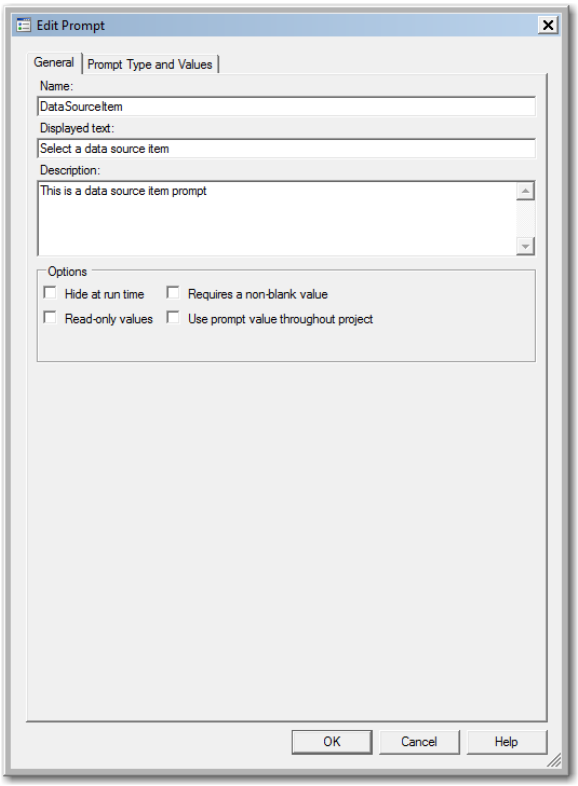
Substituting a SAS Studio Task for a Data Source Prompt

SAS Studio does not have controls with all of the capabilities of a data source prompt. You can create a task with a text input field to prompt for values, or you can use the SAS Studio Task for Data Source Prompt to mold the capability of the `Datasource` control to suit your input needs for a SAS library data source.

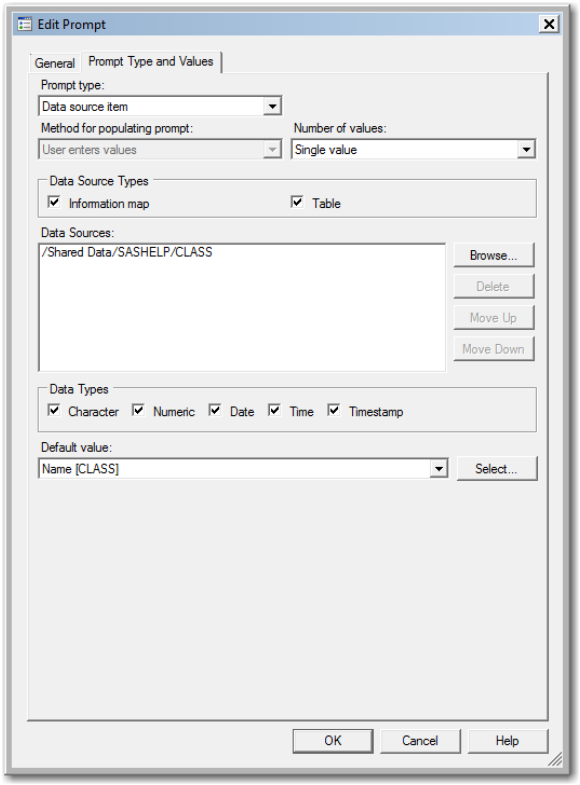
Data Source Item

SAS Enterprise Guide

In this example, a data source item prompt named DataSourceItem is defined as shown in the following two displays.

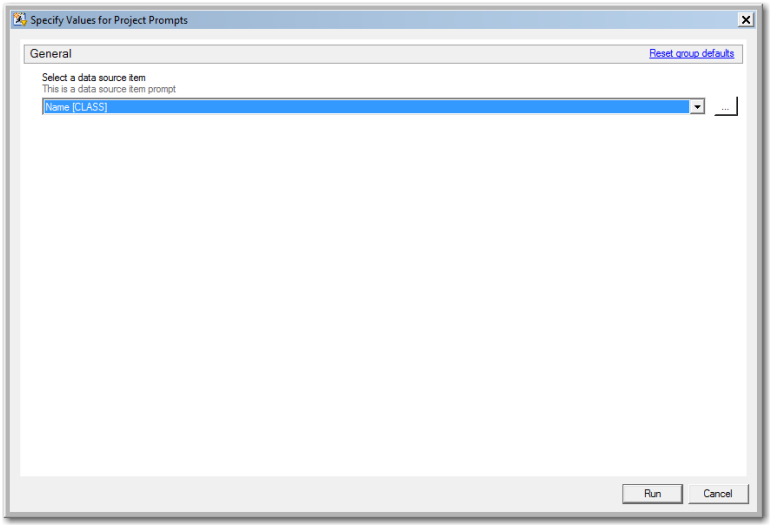


Display 228 - General Properties for Data Source Item Prompt



Display 229 - Type and Values for Data Source Item Prompt

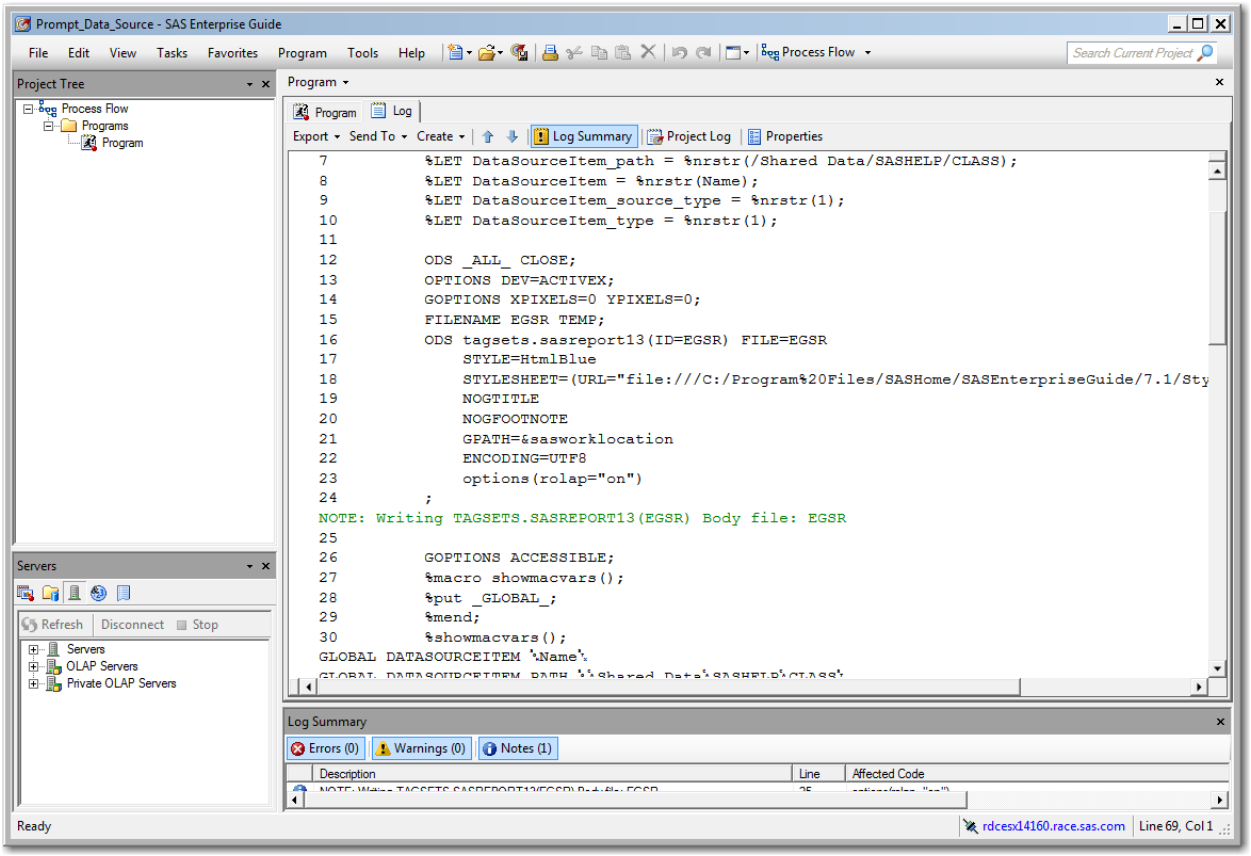
When the code for the prompt is run, the following dialog box appears:



Display 230 - Data Source Item Prompt in Prompt Dialog Box

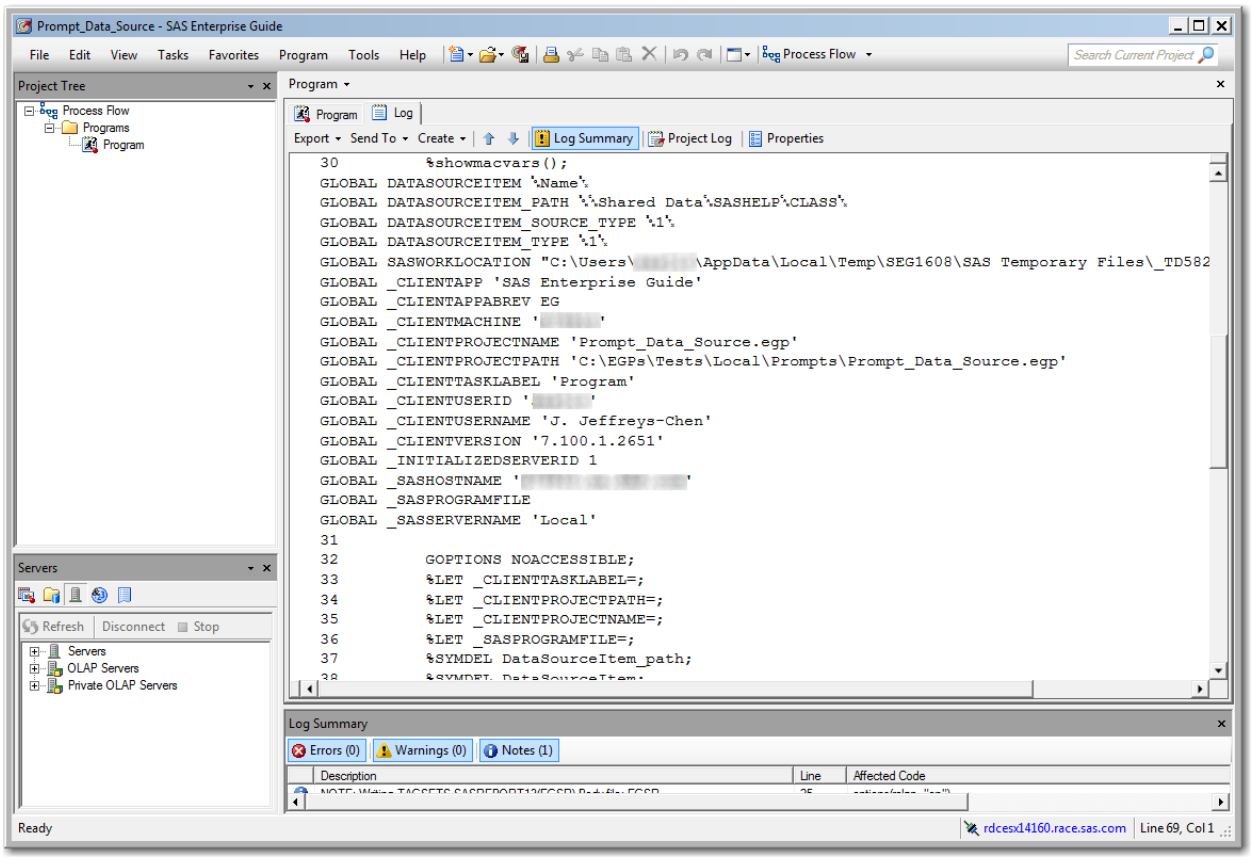
If the user leaves the default value in the data source item prompt field, the following code is generated by SAS Enterprise Guide in the Program node that depends on the prompt.

The %LET statements assign the values specified in the prompt dialog box to the DataSourceItem* macro variables.



Display 231 - %LET Statements for Data Source Item Prompt

Because the **Use prompt value throughout project** is not checked for this prompt, the %SYMDEL statements remove the macro variables at the end of the program.



Display 232 - Values of Global Macro Variables for Data Source Item

SAS Studio

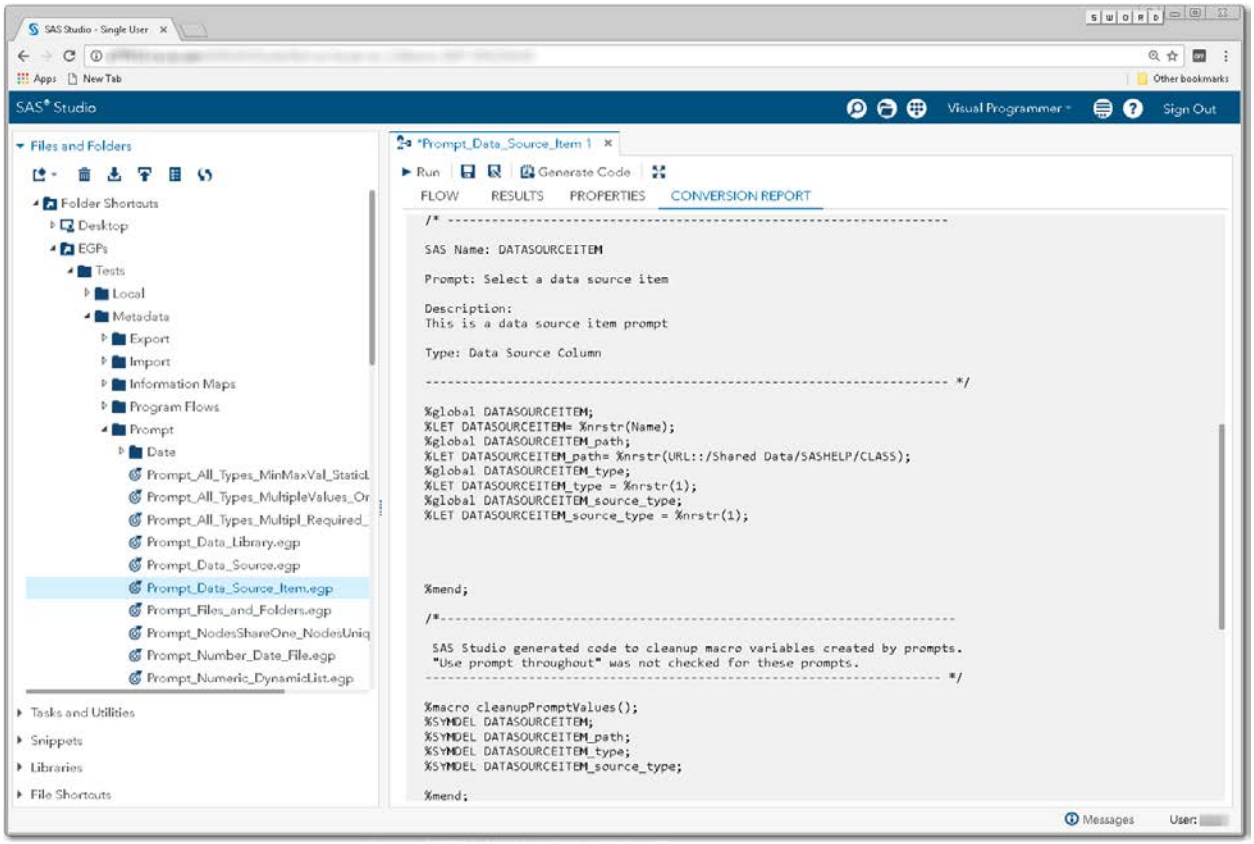
The following display shows code that is added to the converted Program node for the data source item prompt in SAS Enterprise Guide.

These global variables are created:

- DATASOURCEITEM
- DATASOURCEITEM_type
- DATASOURCEITEM_path
- DATASOURCEITEM_source_type

The %LET statements assign the default value to the DATASOURCEITEM* macro variables.

If you want to run your process flow using different values for the DATASOURCEITEM prompt, you must manually update the values of the macro variables in the %LET statements.



Display 233 - Macro Variable Code for Data Source Item Prompt

Note: Currently, the _type macro variables are always set to 1. This value might not be correct for all cases.

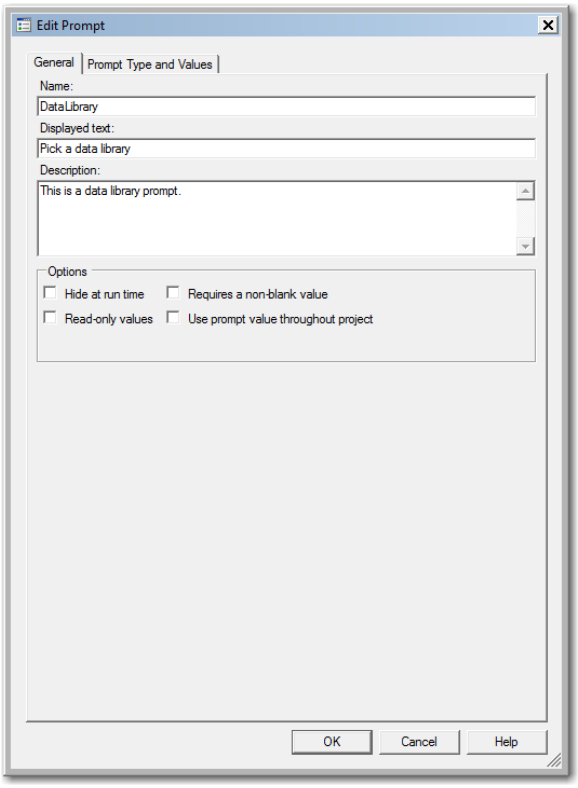
Substituting a SAS Studio Task for a Data Source Item Prompt

SAS Studio does not have controls with all the capabilities of a Data Source Item Prompt. You can create a task that has a text input field to prompt for values, or you can use the SAS Studio Task Data Source Prompt to mold the capability of the `Datasource` control to input a SAS library.

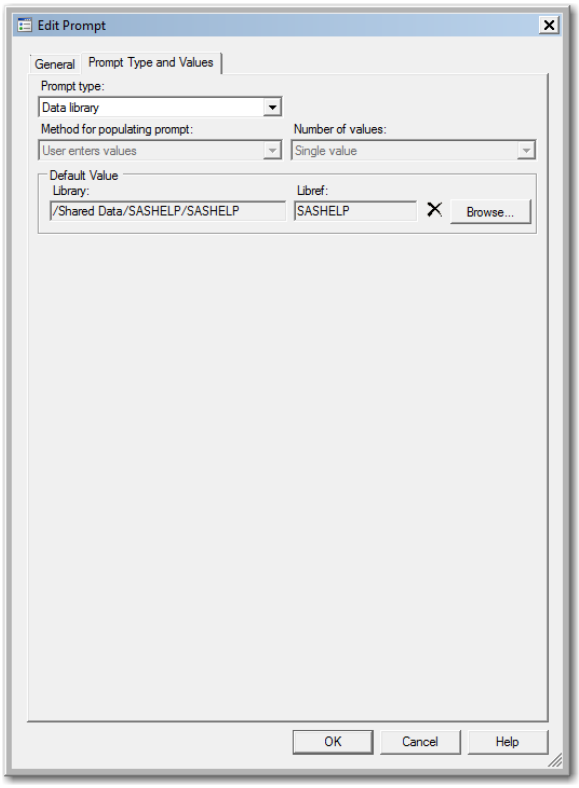
Data Library

SAS Enterprise Guide

In this example, a data library prompt named DataLibrary is defined as shown in the following two displays.

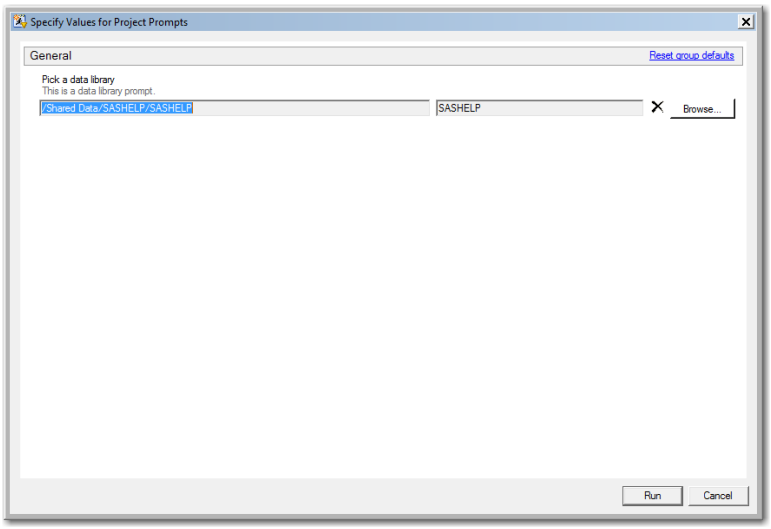


Display 234 - General Properties for Data Library Prompt



Display 235 - Type and Values for Data Library Prompt

When you run the Program node that depends on the prompt, the following dialog box appears:

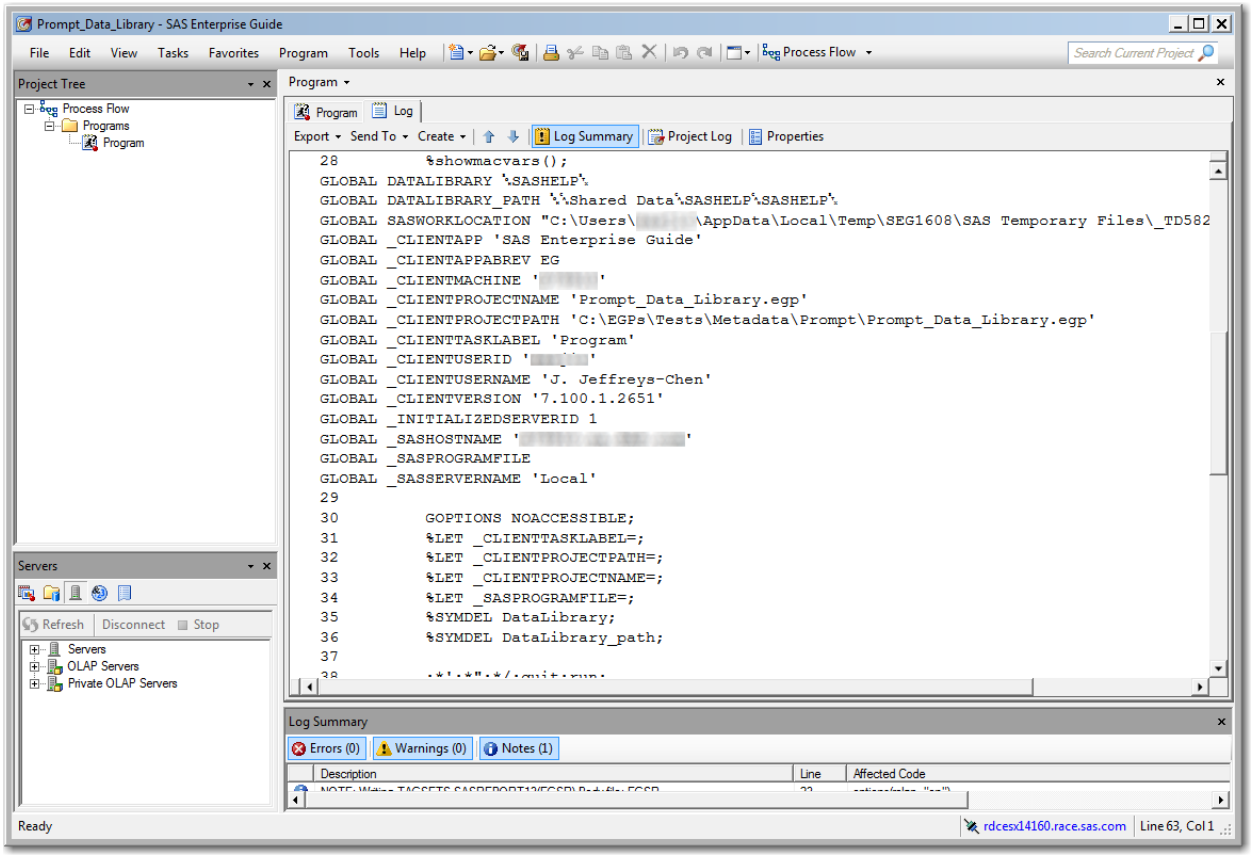


Display 236 - Data Library Prompt in Prompt Dialog Box

If the user leaves the default value in the data library prompt field, the following code is generated by SAS Enterprise Guide in the Program node that depends on the prompt.

The %LET statements assign the values specified in the prompt dialog box to the DataLibrary* macro variables.

Because the **Use prompt value throughout project** is not checked for this prompt, the %SYMDEL statements remove the macro variables at the end of the program.



Display 237 - Global Variables and %SYMDEL Statements for Data Library Prompt

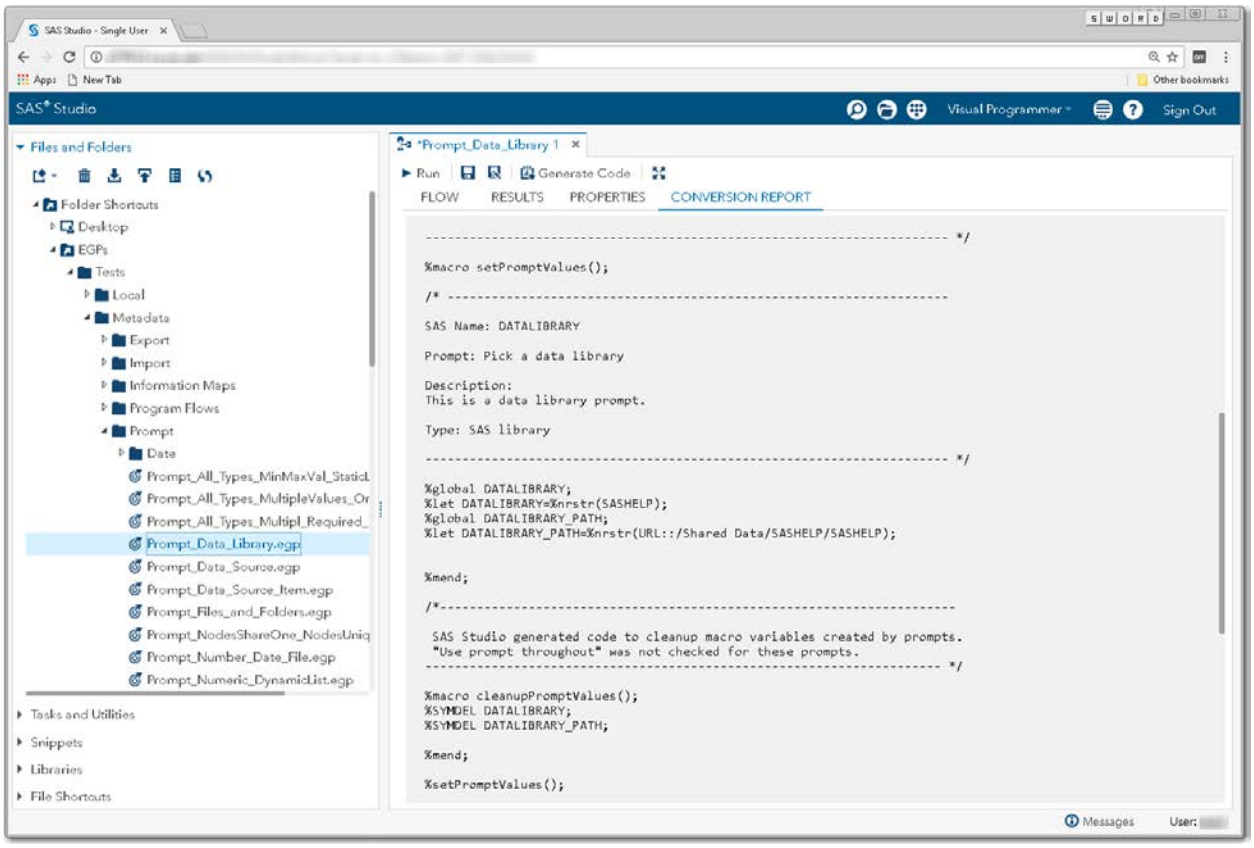
SAS Studio

The following display shows code that is added to the converted Program node for the data library prompt in SAS Enterprise Guide.

These global variables are created:

- DATALIBRARY
- DATALIBRARY_PATH

The %LET statements assign the default value to the DATALIBRARY* macro variables. If you want to run your process flow using different values for the DATALIBRARY prompt, you must manually update the values of the macro variables in the %LET statements.



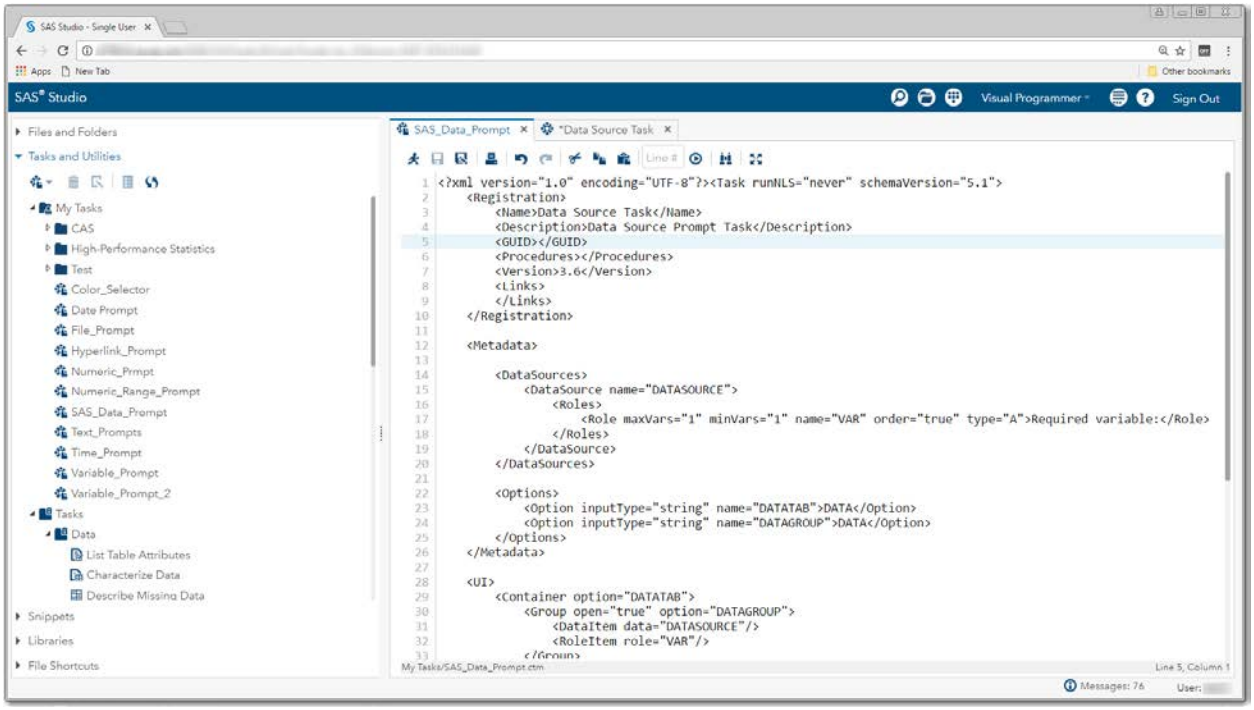
Display 238 - Macro Code for Data Library Prompt

Substituting a SAS Studio Task for a Data Library Prompt

SAS Studio does not have controls with all the capabilities of a data library prompt. You can create a task that has a text input field to prompt for values, or you can use the SAS Studio Task for Data Source Prompt to mold the capability of the `Datasource` control to input a SAS library.

SAS Studio Task for Data Source Prompt

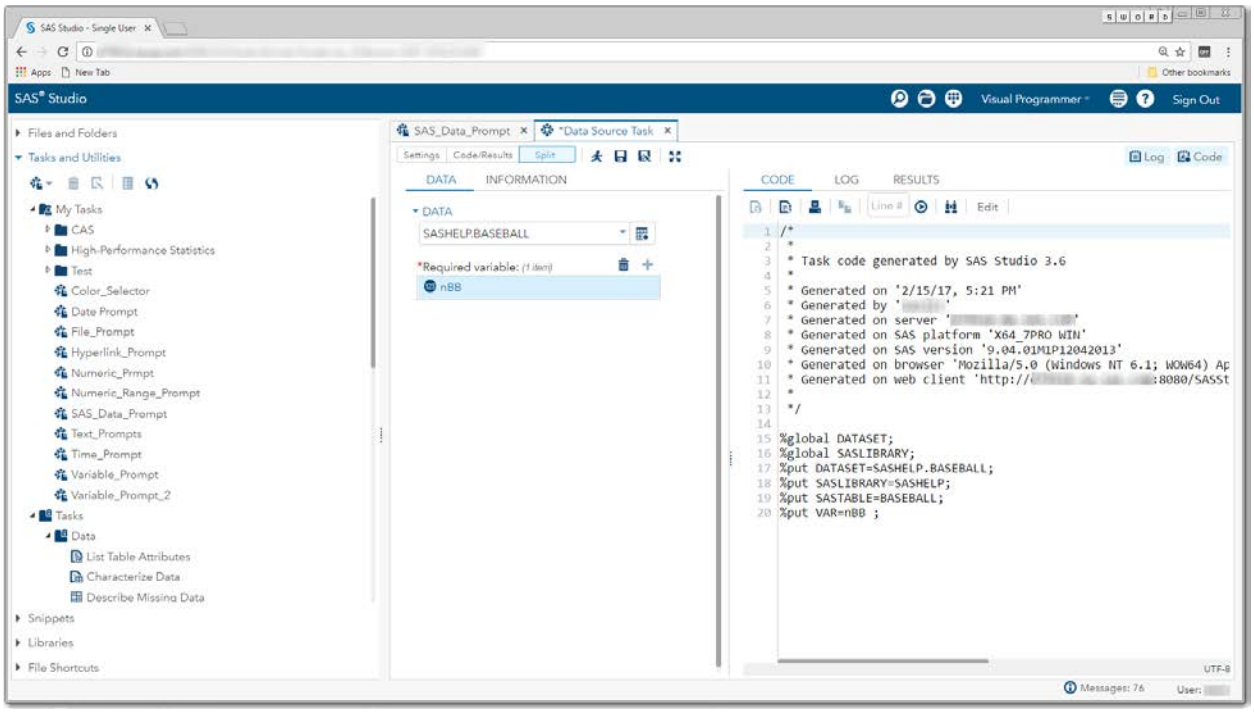
SAS Studio does not have task controls that can be configured to provide all the capabilities in the data source, data source item, and data library prompts in SAS Enterprise Guide. However, SAS Studio does have controls that can be used to prompt for tables and variables. A task that prompts for a library and a table column is shown in the following displays.



Display 239 – Code for SAS_Data_Prompt Task

Here is the complete code for the SAS_Data_Prompt task.

```
<?xml version="1.0" encoding="UTF-8"?><Task runNLS="never" schemaVersion="5.1">
	<Registration>
		<Name>Data Source Task</Name>
		<Description>Data Source Prompt Task</Description>
		<GUID></GUID>
		<Procedures></Procedures>
		<Version>3.6</Version>
		<Links>
		</Links>
	</Registration>

	<Metadata>

		<DataSources>
		    <DataSource name="DATASOURCE">
		      <Roles>
		          <Role maxVars="1" minVars="1" name="VAR" order="true"
		                type="A">
		              Required variable:</Role>
		          </Roles>
		      </DataSource>
		</DataSources>

		<Options>
			<Option inputType="string" name="DATATAB">DATA</Option>
			<Option inputType="string" name="DATAGROUP">DATA</Option>
		</Options>
	</Metadata>

	<UI>
		<Container option="DATATAB">
		    <Group open="true" option="DATAGROUP">
		        <DataItem data="DATASOURCE"/>
		            <RoleItem role="VAR"/>
		    </Group>
		</Container>
	</UI>

	<CodeTemplate>
		<![CDATA[

%global DATASET;
%global SASLIBRARY;
%put DATASET=$DATASOURCE;
%put SASLIBRARY=$DATASOURCE.getLibrary();
%put SASTABLE=$DATASOURCE.getTable();
#if( $VAR.size() > 0 )%put VAR=#foreach( $item in $VAR )$item #end;#end


		]]>
	</CodeTemplate>
</Task>
```

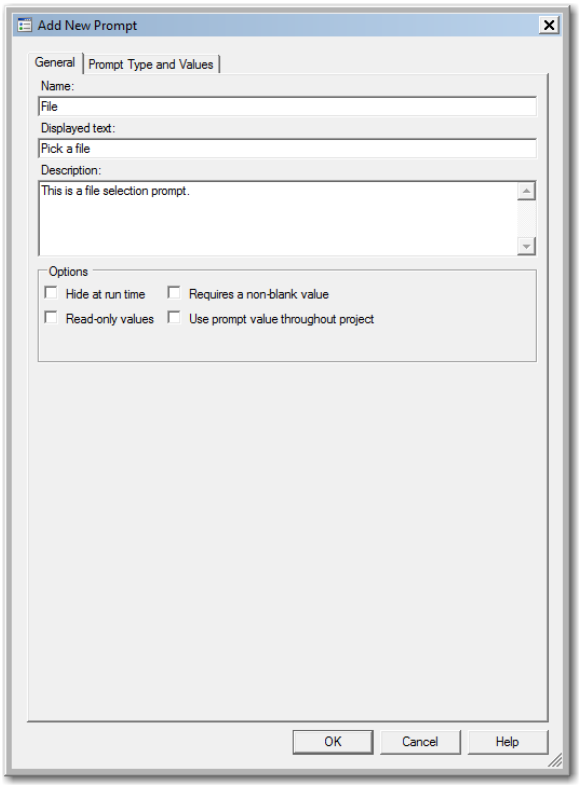


Display 240 - User Interface and Generated SAS Code for Data Source Task

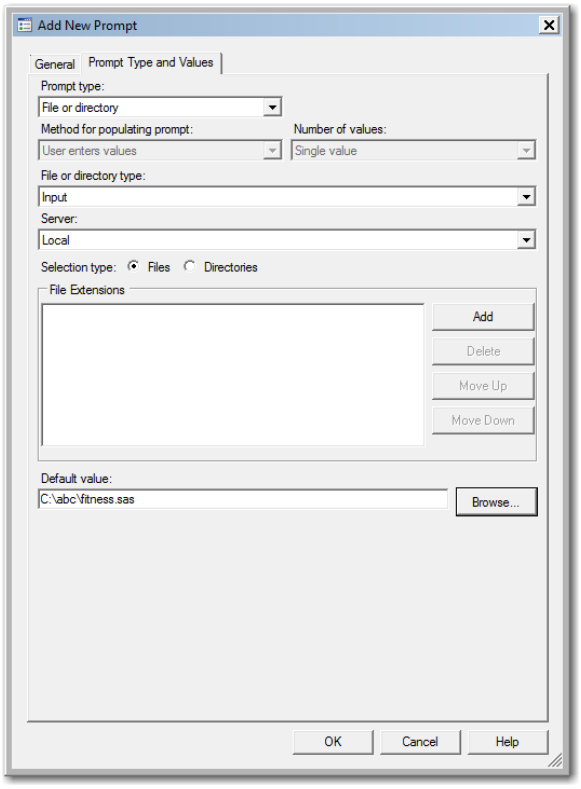
File or Directory

SAS Enterprise Guide

In this example, a file prompt named File is defined as shown in the following two displays.

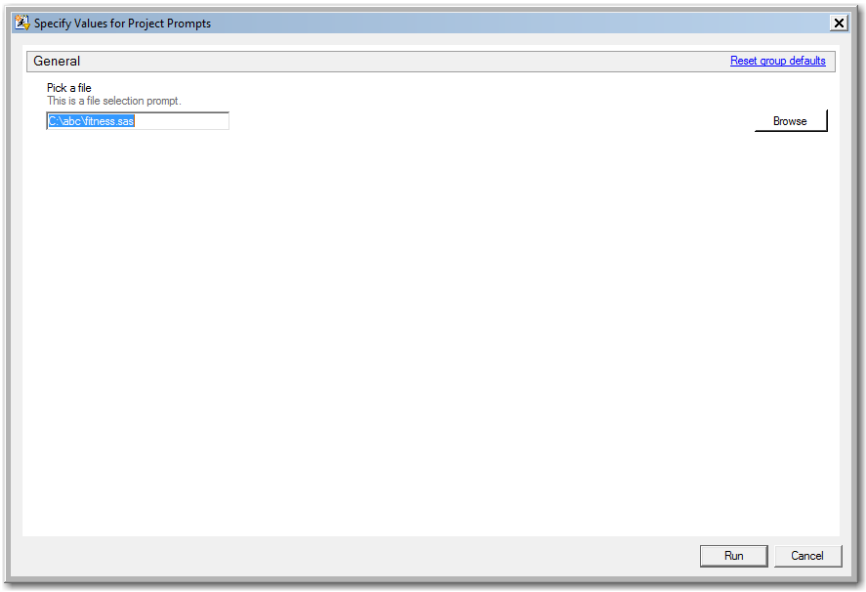


Display 241 - General Properties for File Prompt



Display 242 - Type and Values for File Prompt

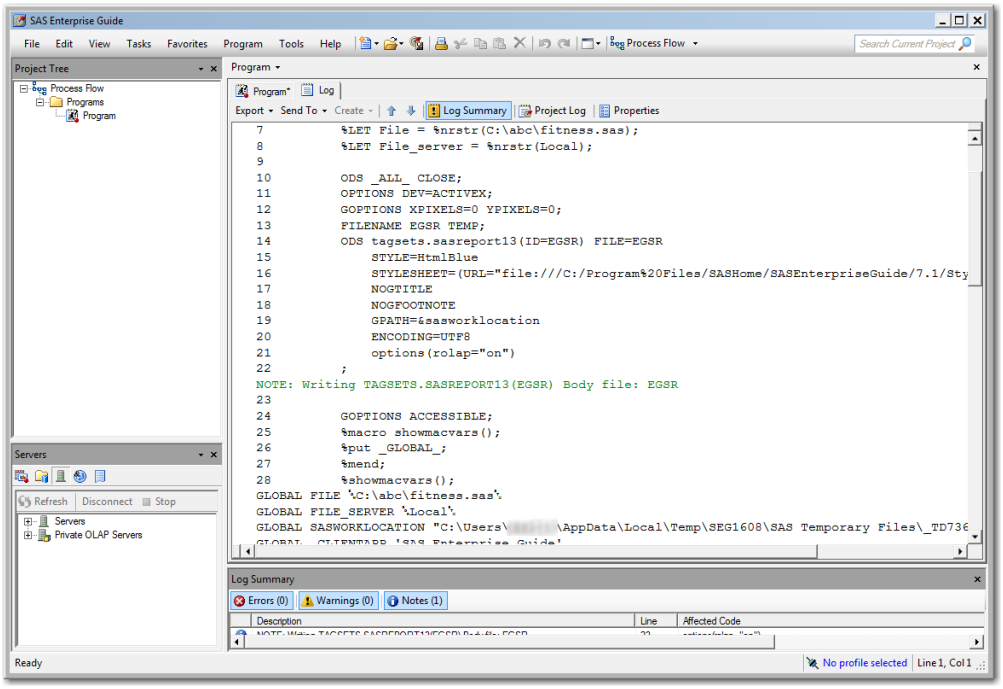
When running the code for this prompt, the following dialog box appears:



Display 243 - File Prompt in Prompt Dialog Box

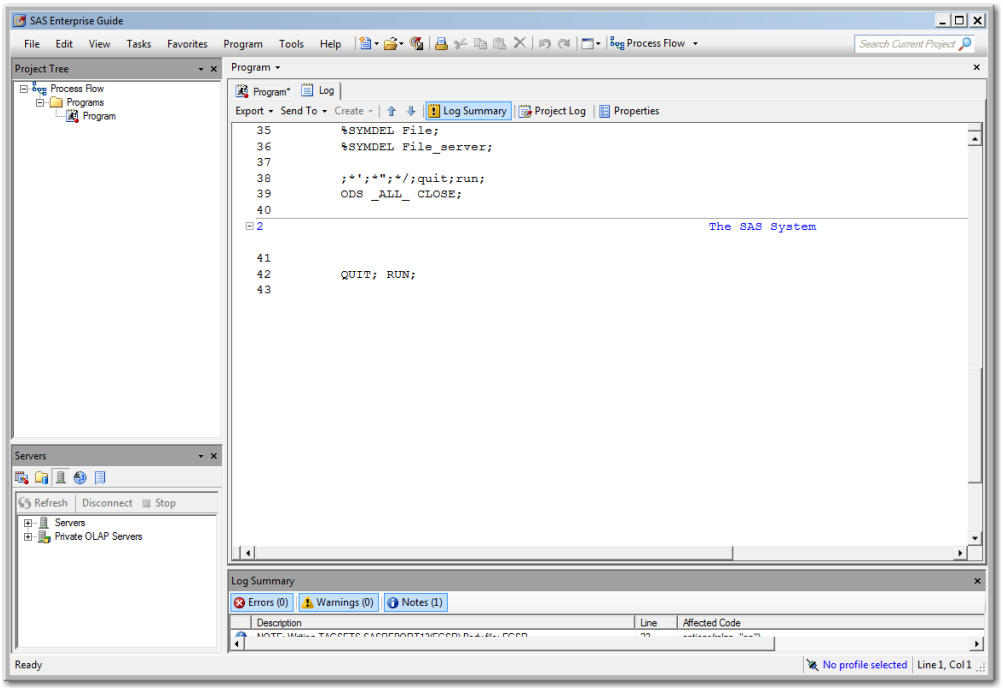
If the user leaves the default value in the file prompt field, the following code is generated by SAS Enterprise Guide in the Program node that depends on the prompt.

The %LET statements assign the values specified in the prompt dialog box to the File* macro variables.



Display 244 - %LET Statements and Values of Global Variables for File Prompt

Because the **Use prompt value throughout project** option is not checked for this prompt, the %SYMDEL statements remove the macro variables at the end of the program.



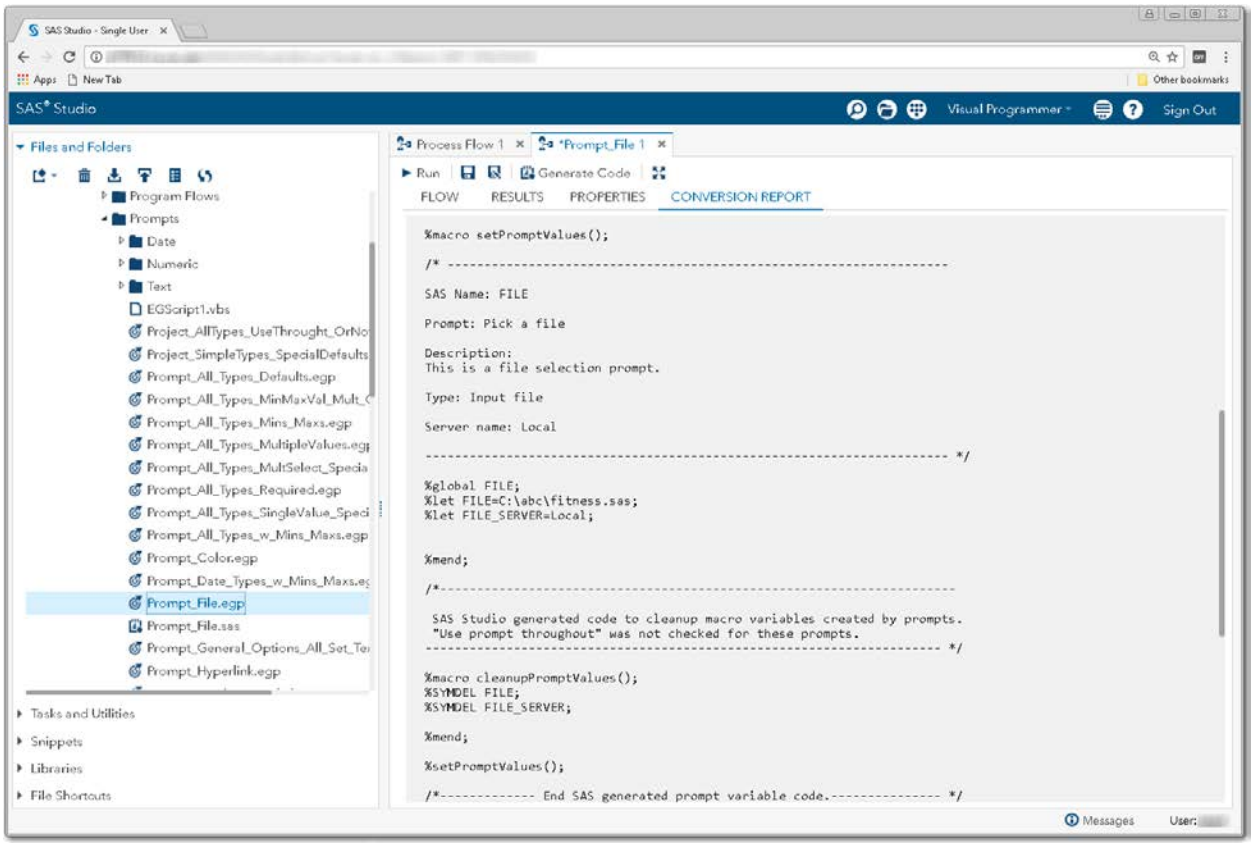
Display 245 - %SYMDEL Statements Remove File* Macro Variables

SAS Studio

The following display shows code that is added to the converted Program node for the file prompt in SAS Enterprise Guide.These global variables are created:

- FILE
- FILE_SERVER

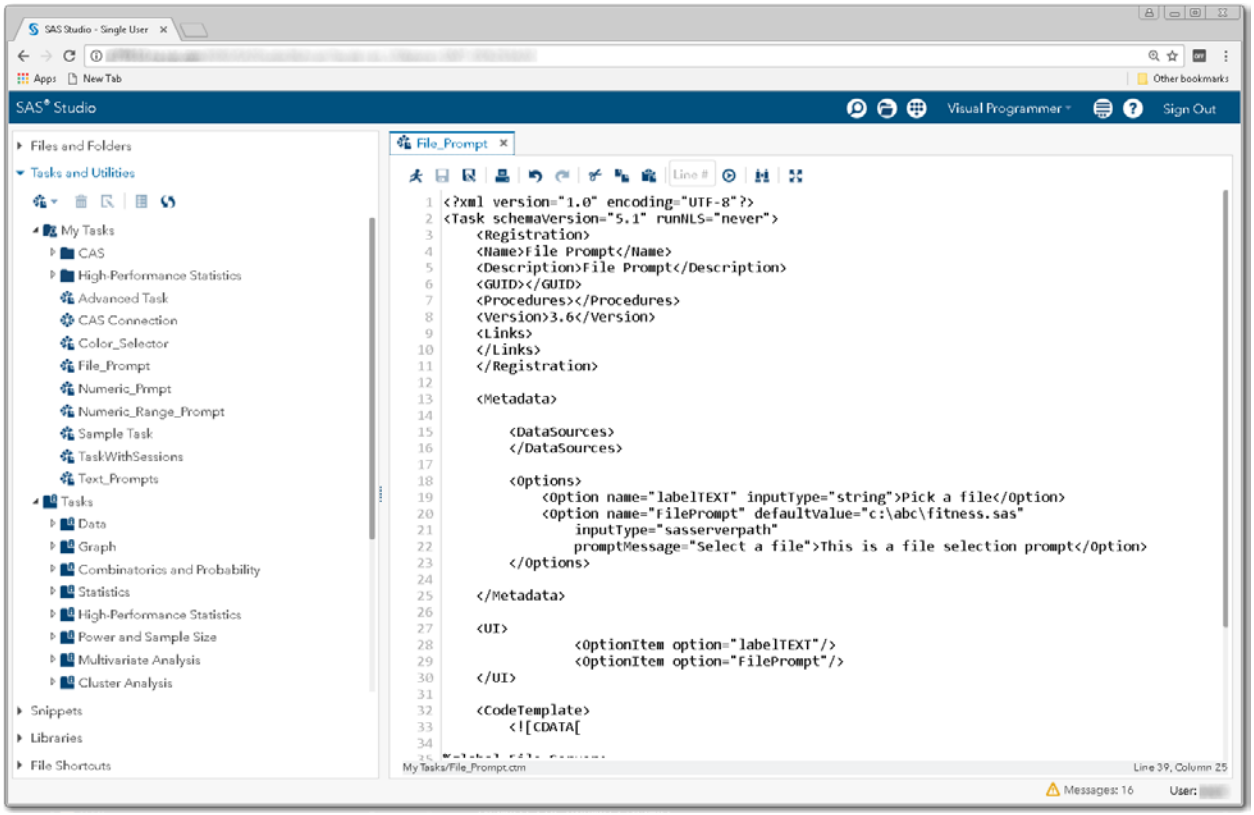
The %LET statements assign the default value to the FILE* macro variables. If you want to run your process flow using different values for the FILE prompt, you must manually update the values of the macro variables in the %LET statements.



Display 246 - Macro Variable Code for the File Prompt

Substituting a SAS Studio Task for a File Prompt

1. Create a SAS Studio task with a control that represents the prompt for a file.
   - Add a label and an sasserverpath input control. Set the name of the input control to FilePrompt.
   - Set the default value to the default value shown in the generated `setPromptValues()` macro in the converted Program node.
   - Change the string of the input control to match the string specified in the prompt.

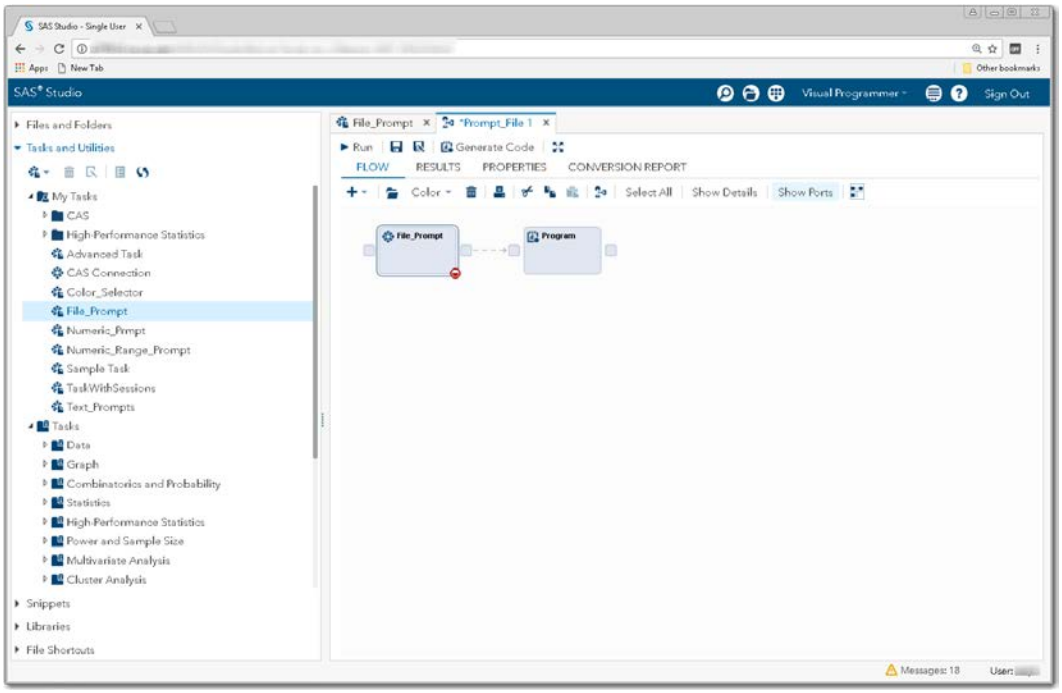


Display 247 - Code for File_Prompt Task

The following code is an example of a task that could be used as the file prompt.

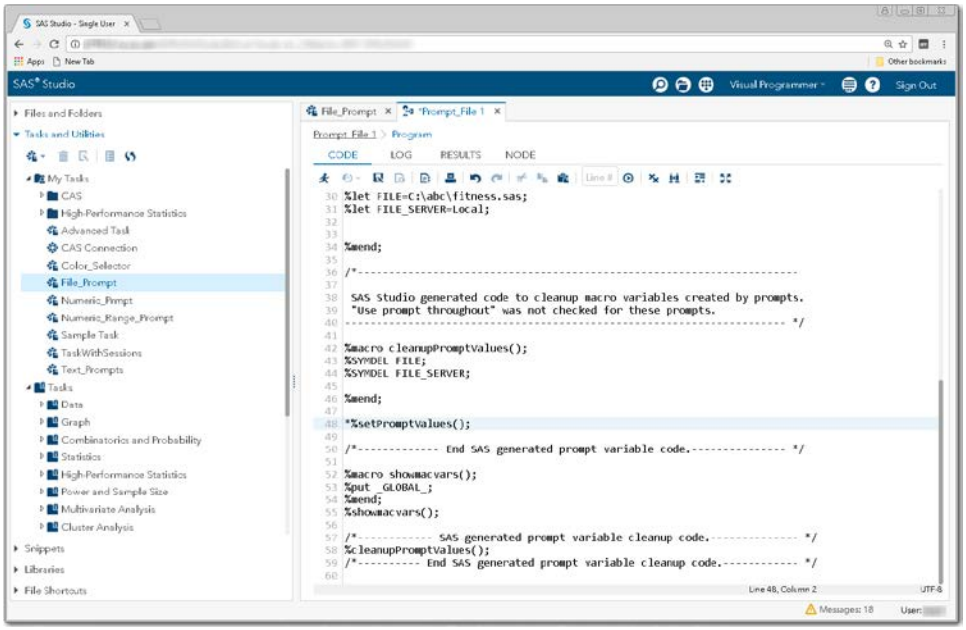
```
<?xml version="1.0" encoding="UTF-8"?>
<Task schemaVersion="5.1" runNLS="never">
	<Registration>
	<Name>File Prompt</Name>
	<Description>File Prompt</Description>
	<GUID></GUID>
	<Procedures></Procedures>
	<Version>3.6</Version>
	<Links>
	</Links>
	</Registration>

	<Metadata>

		<DataSources>
		</DataSources>

		<Options>
			<Option name="labelTEXT" inputType="string">Pick a file</Option>
			<Option name="FilePrompt" defaultValue="c:\abc\fitness.sas"
				inputType="sasserverpath"
				 promptMessage="Select a file">
			    This is a file selection prompt
			</Option>
		</Options>

	</Metadata>

	<UI>
		<OptionItem option="labelTEXT"/>
		<OptionItem option="FilePrompt"/>
	</UI>

	<CodeTemplate>
		<![CDATA[

%global File_Server;
%let File_Server="Local";

%global File;
%let File=${FilePrompt.fullPath};

		]]>
	</CodeTemplate>
</Task>
```

2. Save this task to your **My Tasks** folder.
3. Drag the task from **My Tasks** into your converted process flow.
4. Link the output port of the new task to the input port of the converted Program node.



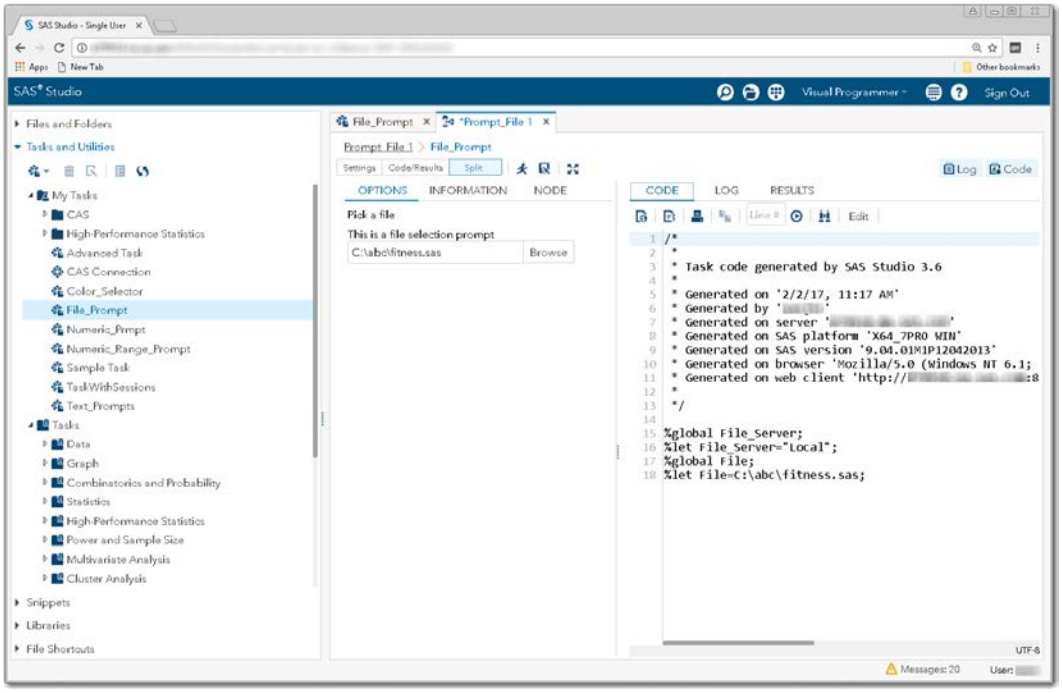
Display 248 - File_Prompt Task Linked to Program Node

5. Comment out the `%setPromptValues()` macro call from the converted Program node. The macro code generated by the File_Prompt task replaces this code.



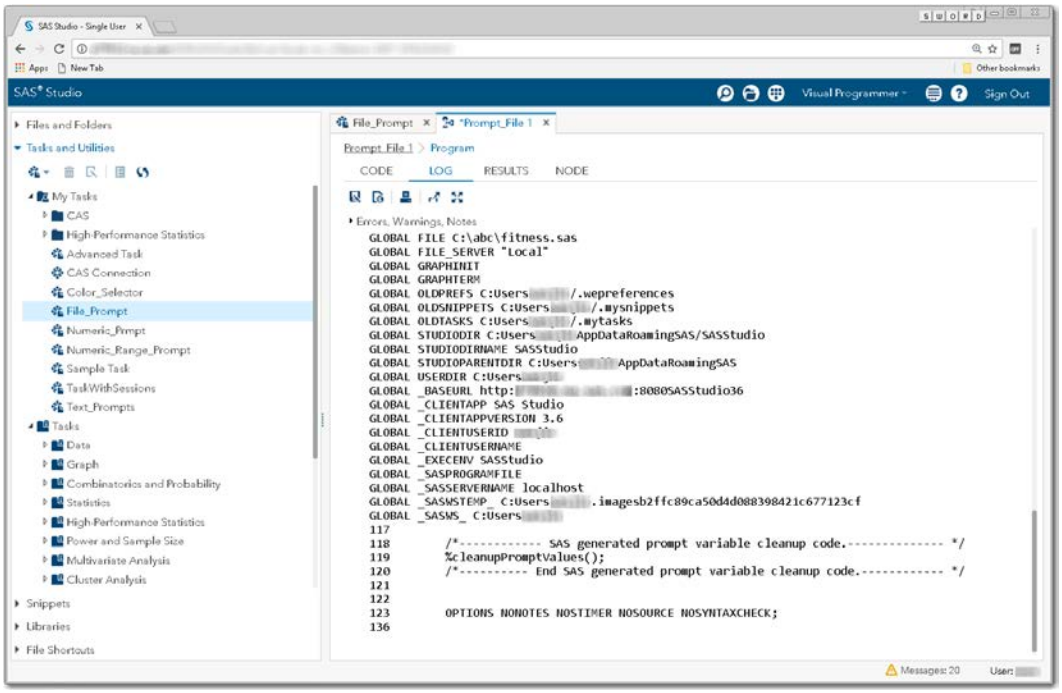
Display 249 - Commented Out %setPromptValues Macro Call

To run your flow with a different value than the default value, open the Prompt_File node and select a different file.



Display 250 - Running the File_Prompt Task

When you run the process flow, the global File* variables are set to the values specified in the task.

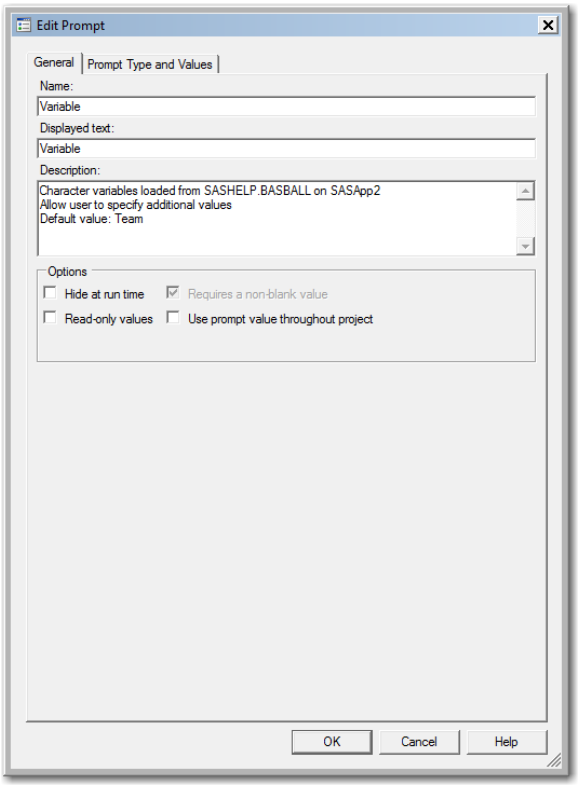


Display 251 - File Prompt Variables with Updated Values

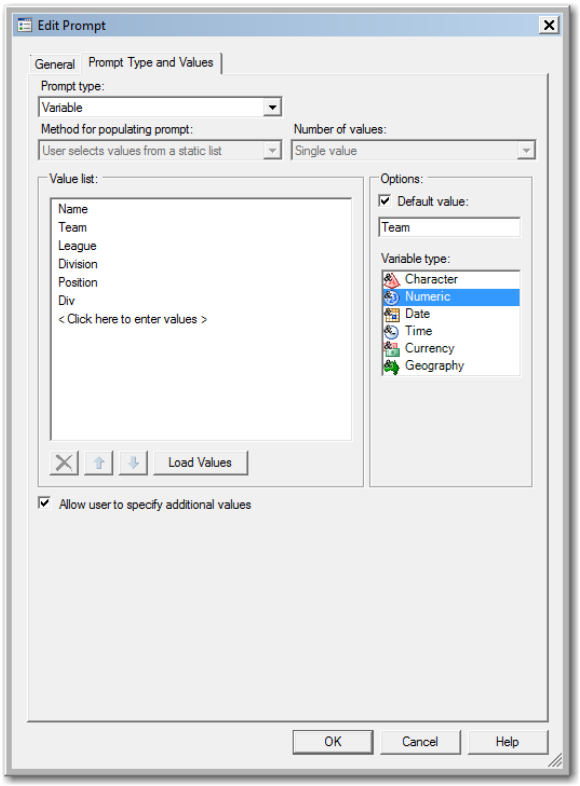
Variable

SAS Enterprise Guide

In this example, a variable prompt named Variable is defined as shown in the following two displays.

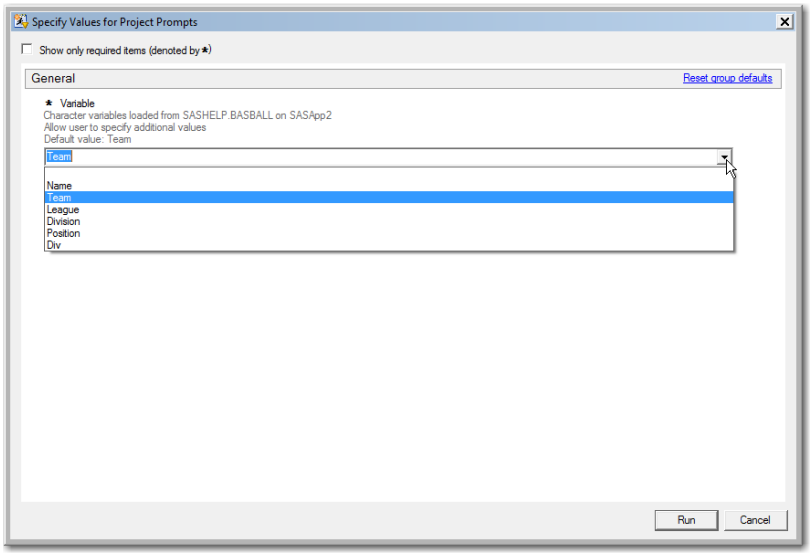


Display 252 - General Properties for Variable Prompt



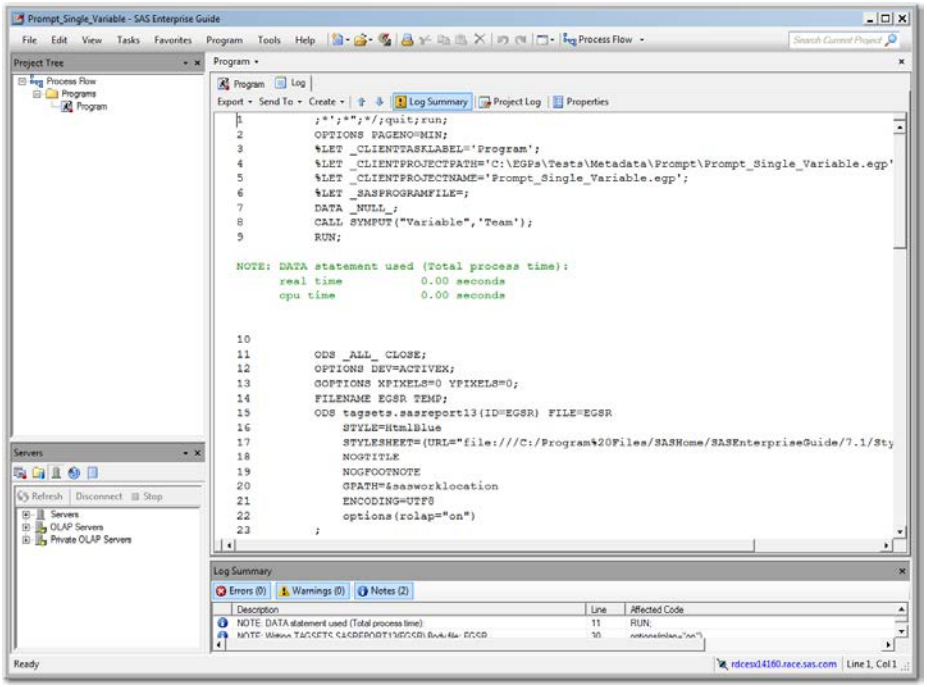
Display 253 - Type and Values for Variable Prompt

When you run the Program node that depends on the prompt, the following display appears:



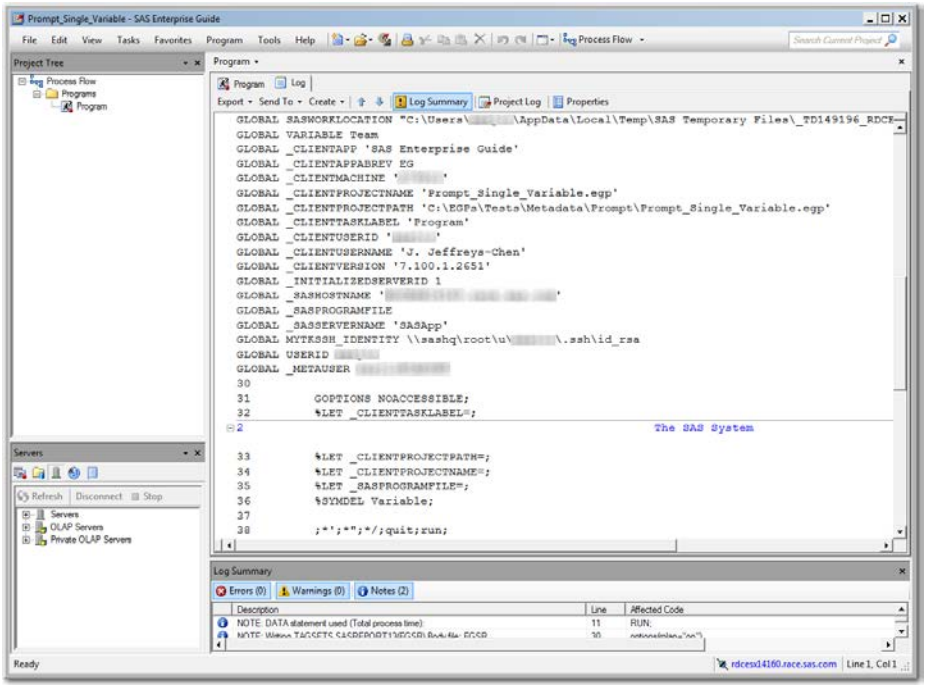
Display 254 – Variable Prompt in the Prompt Dialog Box

If the user leaves the default value in the variable prompt field, the following code is generated by SAS Enterprise Guide in the Program node that depends on the prompt.
The %LET statements assign the value specified in the prompt dialog box to the Variable macro variable.



Display 255 - %LET Statements for Variable Prompt

Because the **Use prompt value throughout project** option is not checked for this prompt, the %SYMDEL statements remove the macro variables at the end of the program.

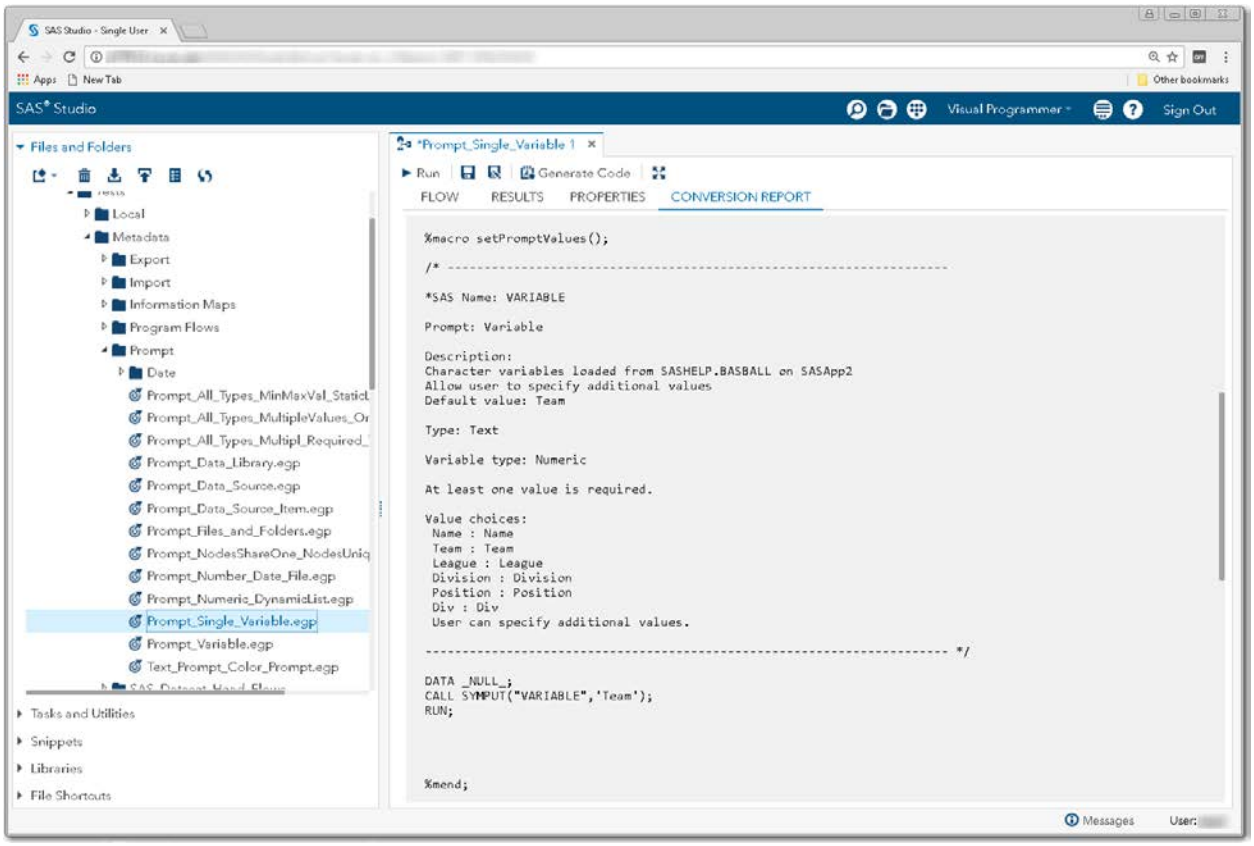


Display 256 – Values of Global Variables and %SYMDEL Statements for Variable Prompt

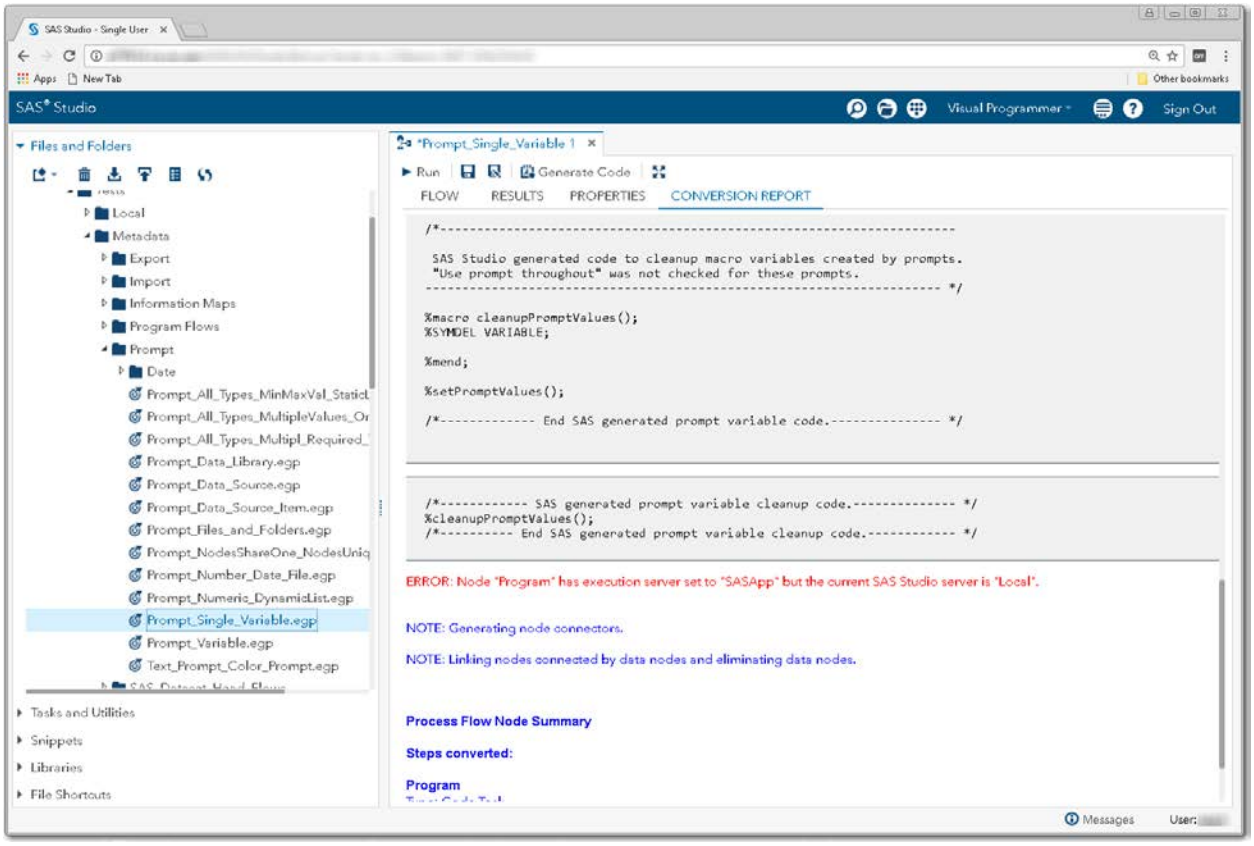
SAS Studio

The following display shows code that is added to the converted Program node for the variable prompt in SAS Enterprise Guide.

The global variable VARIABLE is created, and the %LET statement assigns the default value to the VARIABLE macro variable. If you want to run the process flow using a different value for the VARIABLE prompt, you must manually update the value of the macro variable in the %LET statement.



Display 257 - %SYMPUT Code for Variable Prompt

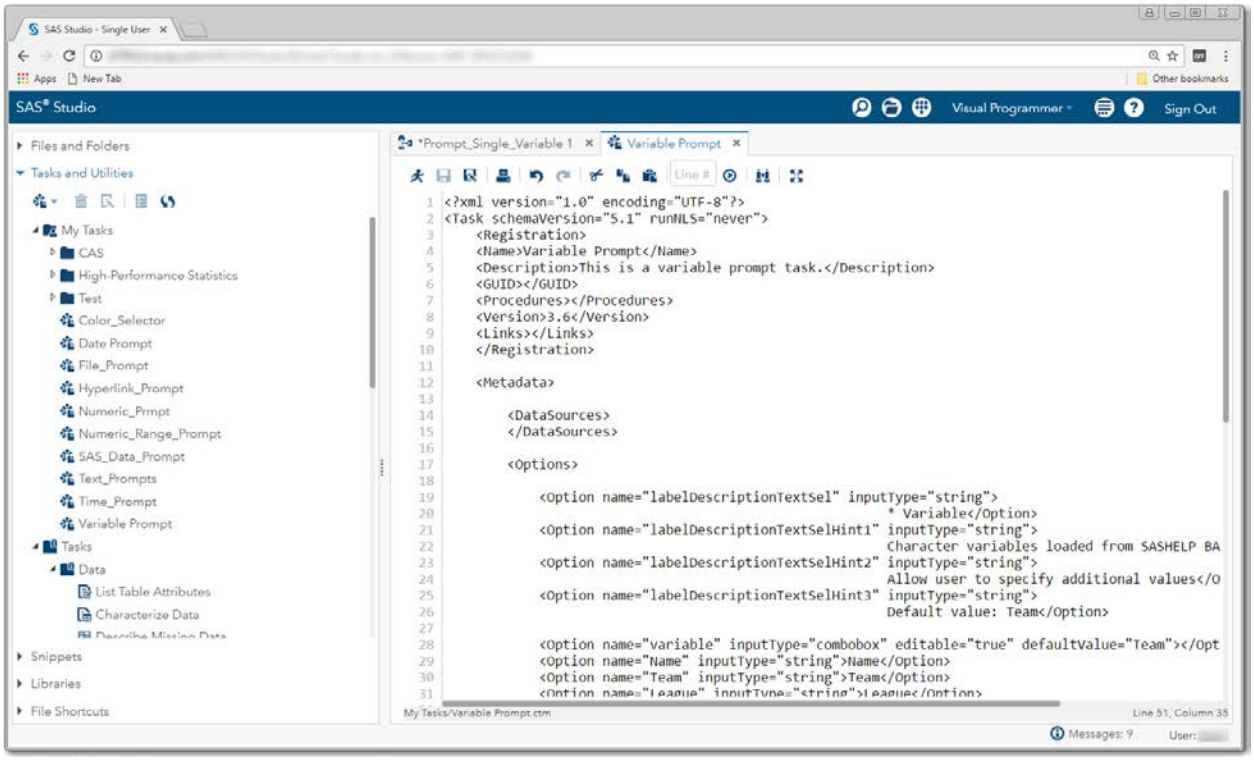


Display 258 - %SYMDEL Code for Variable Prompt

Substituting a SAS Studio Task for Variable Prompt

When defining a variable prompt in SAS Enterprise Guide, the value choices for the prompt are gathered from a data source. These values are then available in a drop-down list when the prompt is displayed. So to replace this type of prompt in a SAS Studio task, you can simply use the combobox control.

1. Create a SAS Studio task with a control that represents the prompt for a Variable.
    - Add controls as shown in the Variable Prompt task.
    - Set the default value to the default value shown in the generated `setPromptValues()` macro in the converted Program node.
    - Change the strings of the input controls to match the strings specified in the prompt.



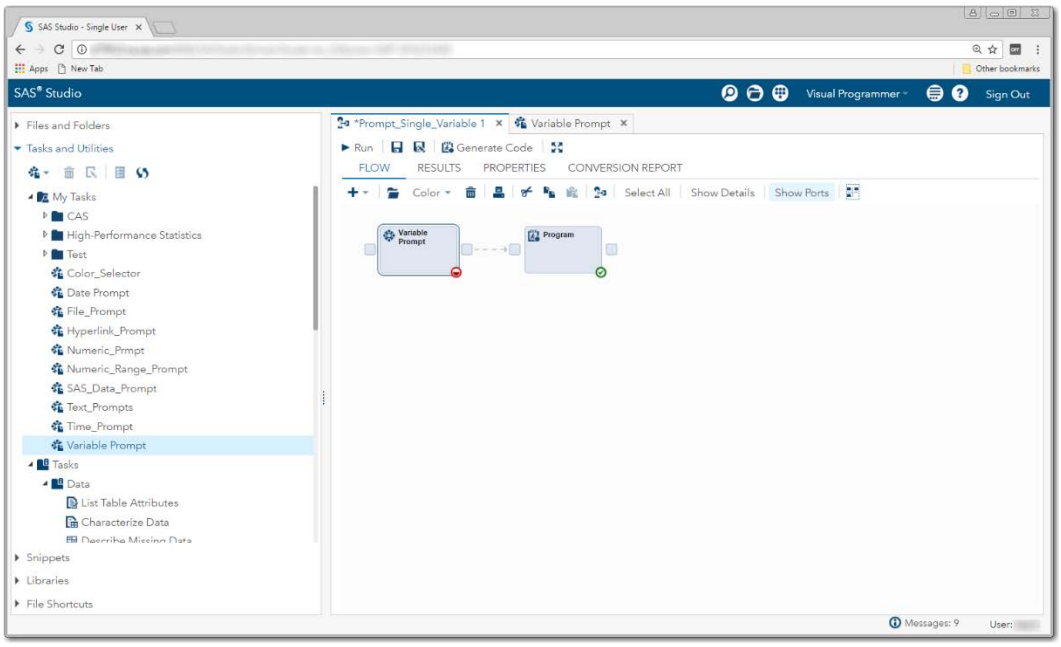
Display 259 - Replacement Task for Variable Prompt

The following code is an example of a task that could be used as the variable prompt.

```
<?xml version="1.0" encoding="UTF-8"?>
<Task schemaVersion="5.1" runNLS="never">
       <Registration>
       <Name>Variable Prompt</Name>
       <Description>This is a variable prompt task.</Description>
       <GUID></GUID>
       <Procedures></Procedures>
       <Version>3.6</Version>
       <Links></Links>
       </Registration>
       <Metadata>
              <DataSources>
              </DataSources>

              <Options>
                     <Option name="labelDescriptionTextSel" inputType="string">
                          * Variable
                     </Option>

                     <Option name="labelDescriptionTextSelHint1" inputType="string">
                         Character variables loaded from SASHELP BASBALL on SASApp2
                     </Option>

                     <Option name="labelDescriptionTextSelHint2" inputType="string">
                         Allow user to specify additional values
                     </Option>

                     <Option name="labelDescriptionTextSelHint3" inputType="string">
                          Default value: Team
                     </Option>

                     <Option name="variable" inputType="combobox" editable="true"
                             defaultValue="Team">
                     </Option>

                     <Option name="Name" inputType="string">Name</Option>
                     <Option name="Team" inputType="string">Team</Option>
                     <Option name="League" inputType="string">League</Option>
                     <Option name="Division" inputType="string">Division</Option>
                     <Option name="Position" inputType="string">Position</Option>
                     <Option name="Div" inputType="string">Div</Option>
              </Options>
       </Metadata>

       <UI>
              <OptionItem option="labelDescriptionTextSel"/>
              <OptionItem option="labelDescriptionTextSelHint1"/>
              <OptionItem option="labelDescriptionTextSelHint2"/>
              <OptionItem option="labelDescriptionTextSelHint3"/>
                             <OptionChoice option="variable">
                     <OptionItem option="Name"/>
                     <OptionItem option="Team"/>
                     <OptionItem option="League"/>
                     <OptionItem option="Division"/>
                     <OptionItem option="Position"/>
                     <OptionItem option="Div"/>
              </OptionChoice>
       </UI>
```

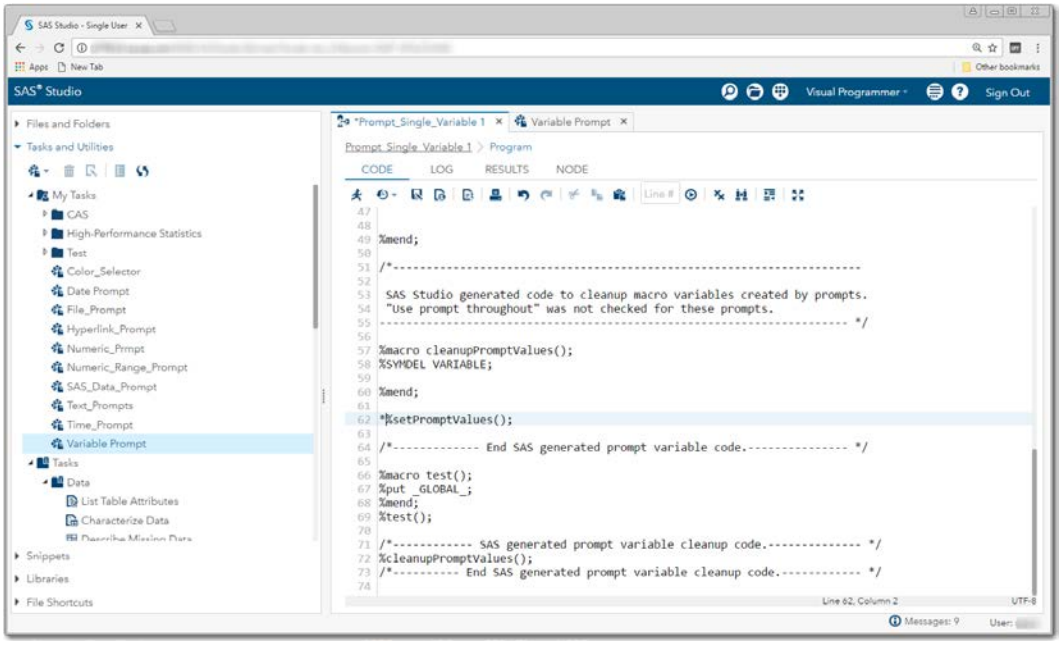
```
<CodeTemplate>
             <![CDATA[
DATA _NULL_;
   CALL SYMPUT("Variable", '${variable}');
run;
             ]]>
       </CodeTemplate>
</Task>
```

2. Save the prompt replacement task to your **My Tasks** folder.
3. Drag the task from **My Tasks** into your converted process flow.
4. Link the output port of the Variable Prompt task to the input port of the converted Program node.



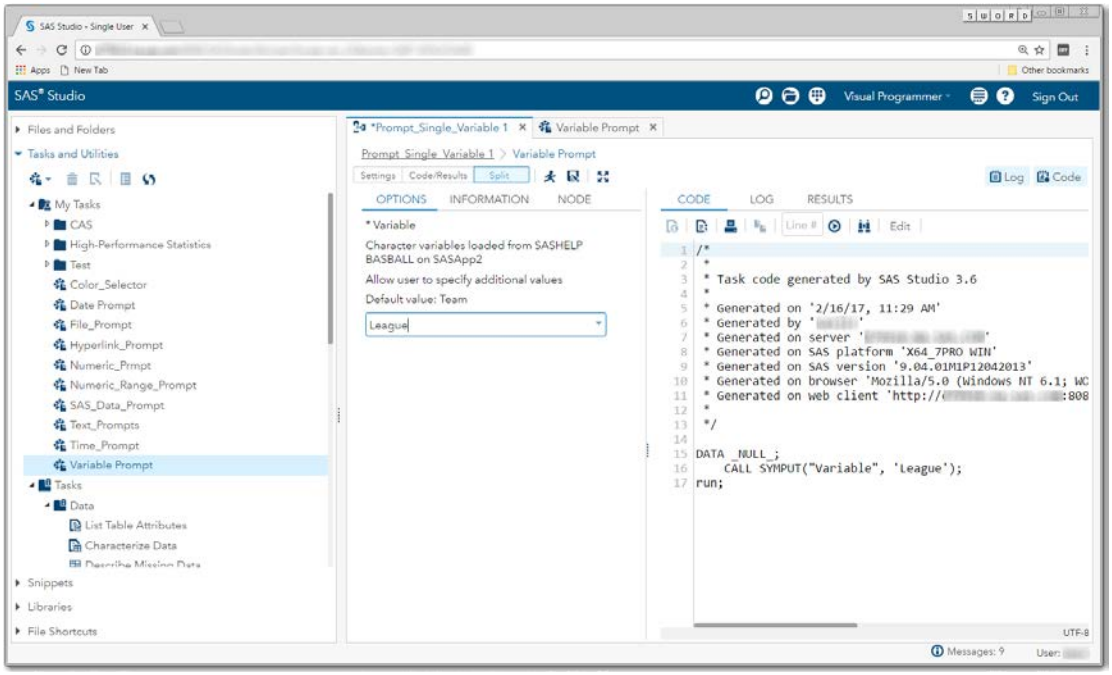
Display 260 – Variable Prompt Task Linked to Program Node

5. Comment out the `%setPromptValues()` macro call from the converted Program node. The macro code generated by the Variable Prompt task replaces this code.



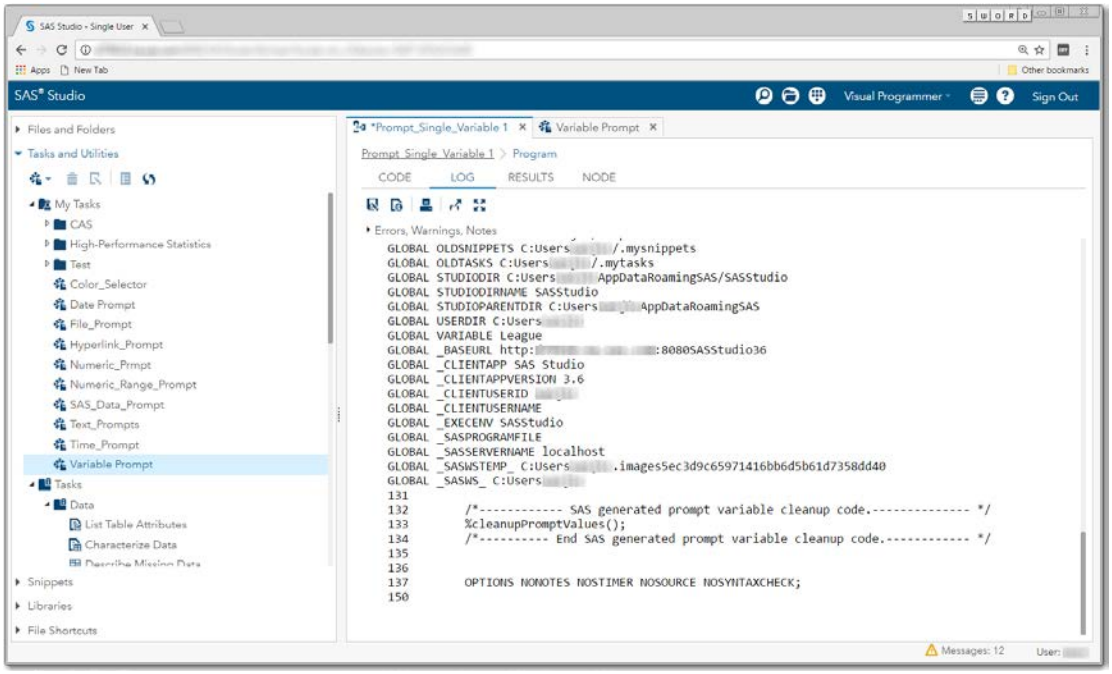
Display 261 – Commented out %setPromptValues in Program Node

To run your flow with a different variable than the default value, open the Variable Prompt node and specify a different value.



Display 262 – User Interface and Generated Code for Variable Prompt Task

When you run the process flow, the global Variable macro variable is set to the value specified in the task.

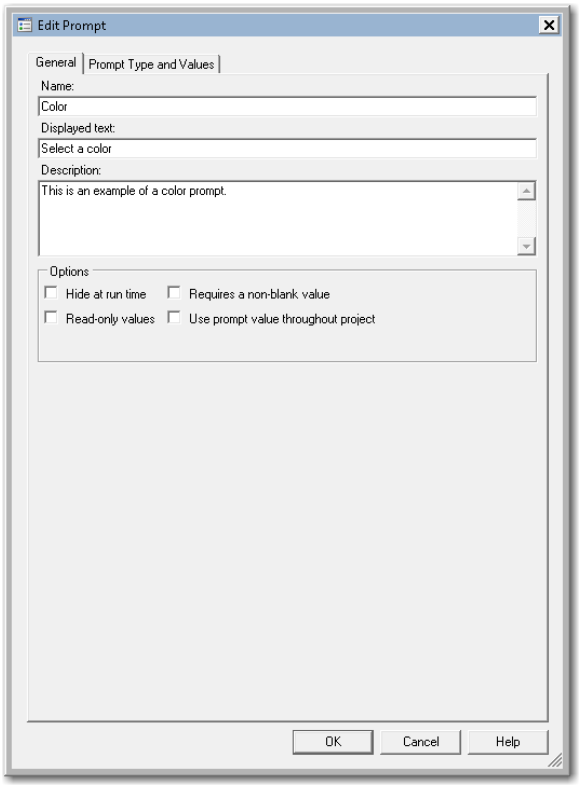


Display 263 – Variable Prompt Variable with Updated Values

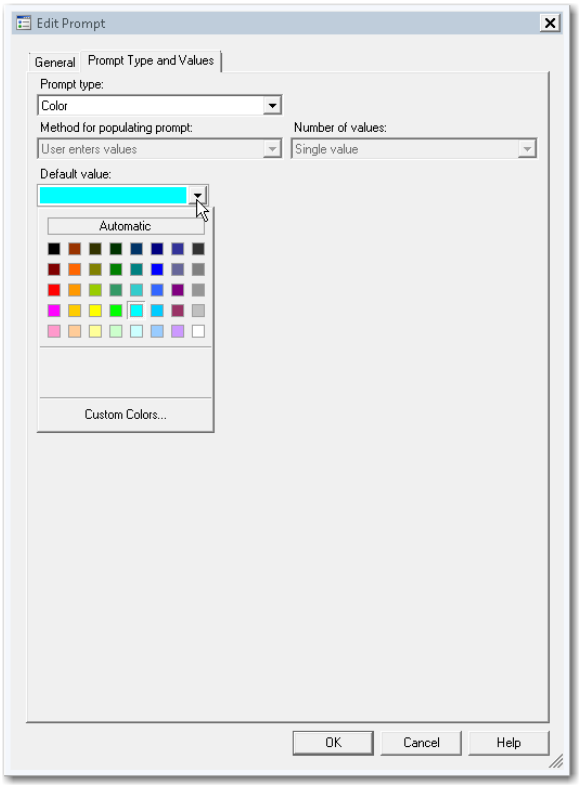
Color

SAS Enterprise Guide

In this example, a color prompt named Color is defined as shown in the following two displays.

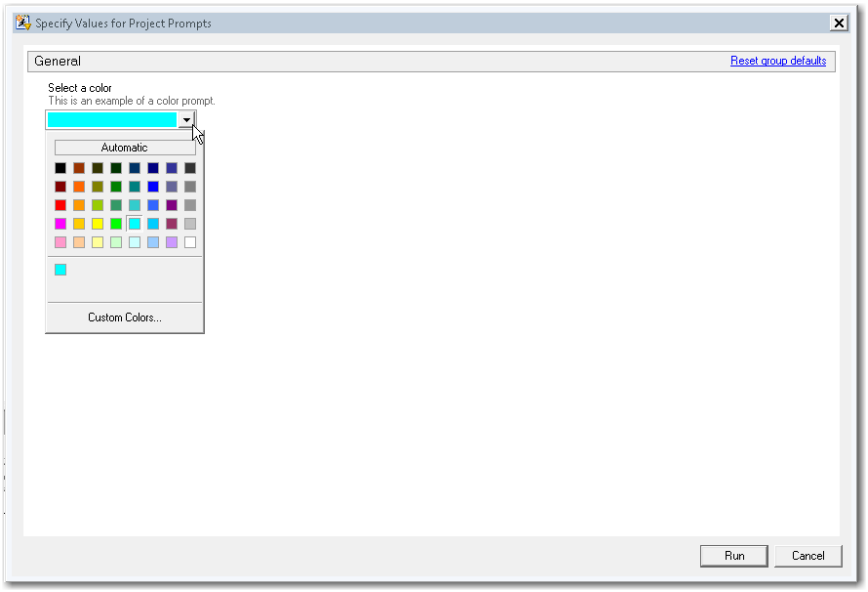


Display 264 - General Properties for Color Prompt



Display 265 - Type and Values for Color Prompt

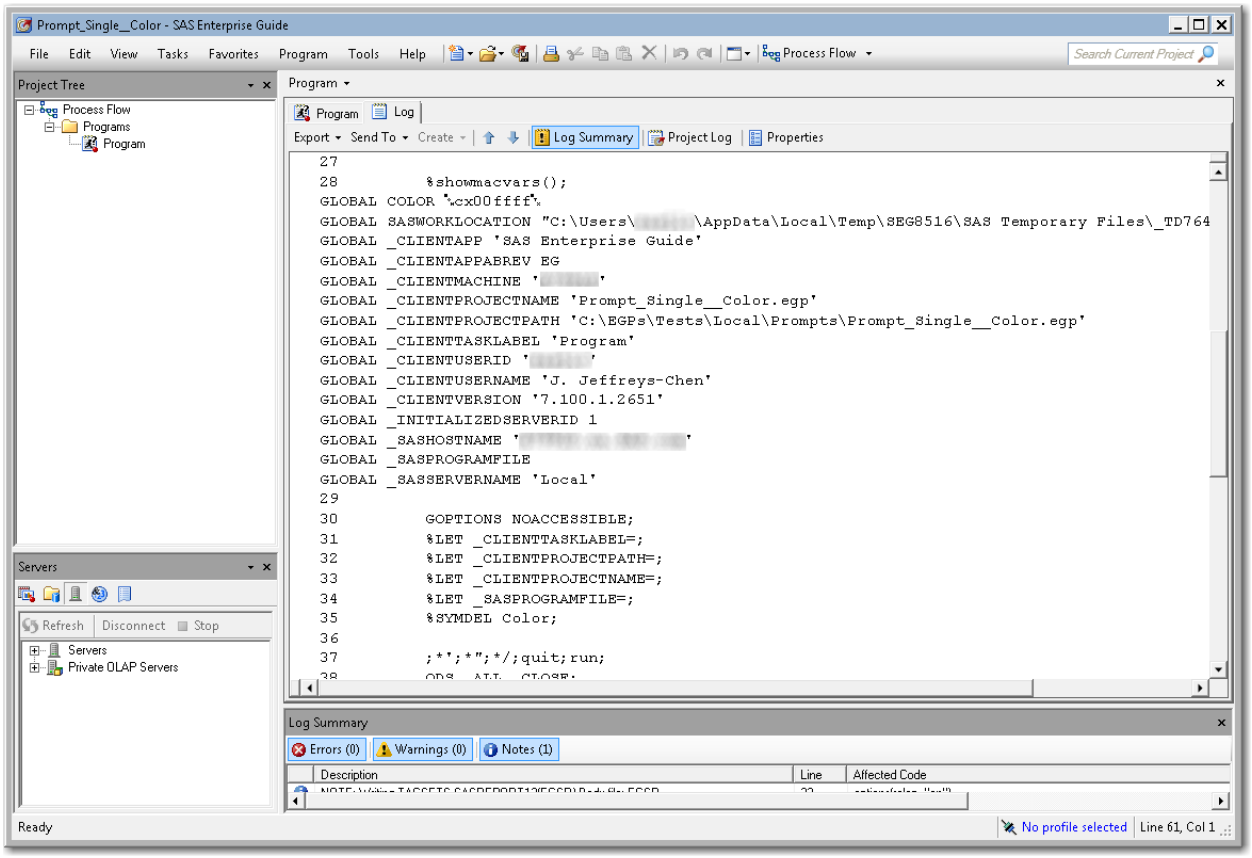
When you run the Program node that depends on the prompt, the following dialog box appears:



Display 266 - Color Prompt in the Prompt Dialog Box

If the user leaves the default value in the color prompt, the following macro variable is generated by SAS Enterprise Guide in the Program node that depends on the prompt.

Because the **Use prompt value throughout project** option is not checked for this prompt, the %SYMDEL statements remove the macro variables at the end of the program.

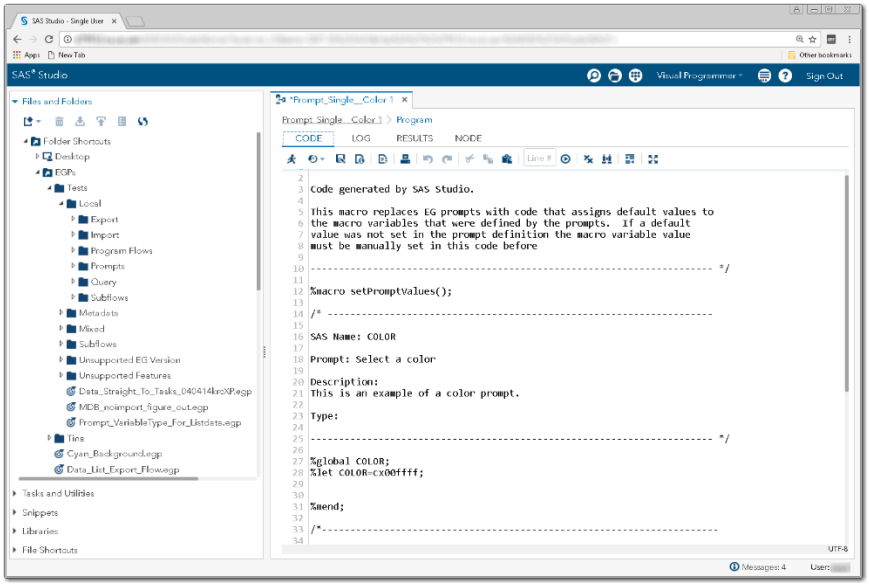


Display 267 – Values of GLOBAL Variables and and %SYMDEL Statement for Color Prompt

SAS Studio

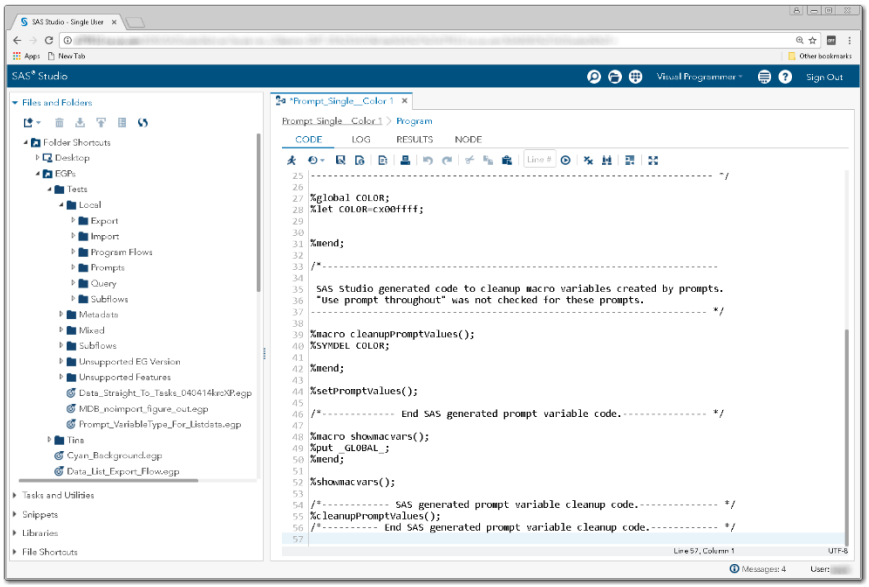
The following display shows code that is added to the converted Program node for the color prompt in SAS Enterprise Guide.

A global variable named COLOR is created, and a %LET statement assigns the default value to COLOR. If you want to run your process flow using a different value for the COLOR prompt, you must manually update the value of the macro variable in the %LET statement.



Display 268 - Code for Color Prompt

Because the **Use prompt value throughout project** option is not checked, the %SYMDEL statement removes the COLOR macro variable.



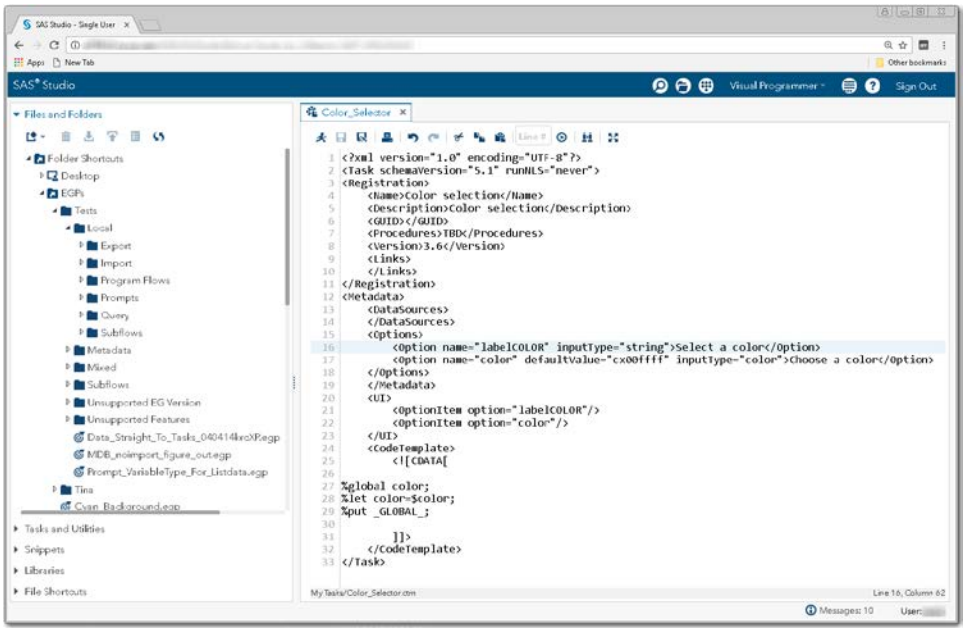
Display 269 - %SYMDEL Code Removes the Color Macro Variable

Substituting a SAS Studio Task for a Color Prompt

1. Create a SAS Studio task with a control that represents the color prompt.
    - Add a label and a color input control. Enter Color as the name of the input control.
    - Set the default value to the default value shown in the `generated setPromptValues()` macro in the converted Program node.
    - Change the string for the input control to **Select a color.**

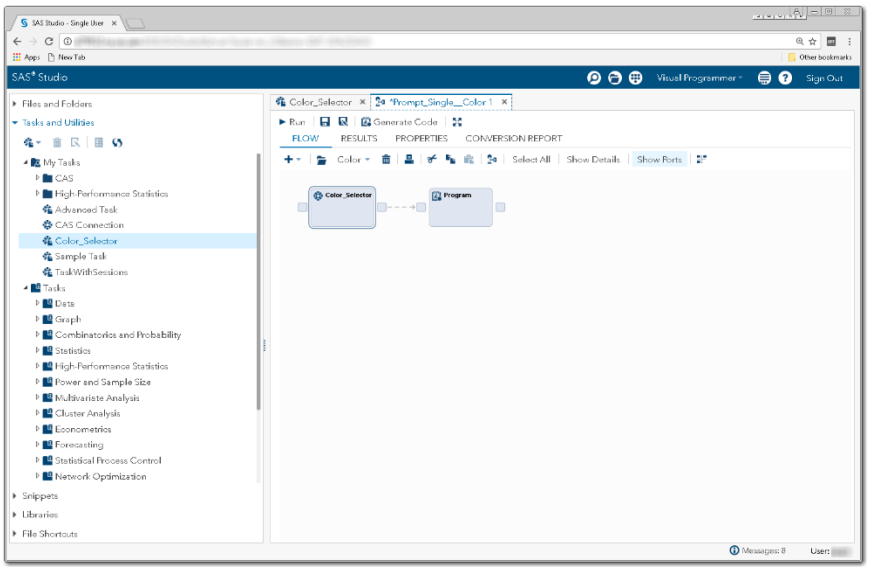
The following code is an example of a task that could be used as the color prompt.

```
<?xml version="1.0" encoding="UTF-8"?>
<Task schemaVersion="5.1" runNLS="never">
<Registration>
       <Name>Color selection</Name>
       <Description>Color selection</Description>
       <GUID></GUID>
       <Procedures>TBD</Procedures>
       <Version>3.6</Version>
       <Links>
       </Links>
</Registration>
<Metadata>
       <DataSources>
       </DataSources>
       <Options>
              <Option name="labelCOLOR" inputType="string">Select a color</Option>
              <Option name="color" defaultValue="cx00ffff" inputType="color">
                  Choose a color
             </Option>
       </Options>
       </Metadata>
       <UI>
              <OptionItem option="labelCOLOR"/>
              <OptionItem option="color"/>
       </UI>
       <CodeTemplate>
              <![CDATA[

%global color;
%let color=$color;
              ]]>
       </CodeTemplate>
</Task>
```



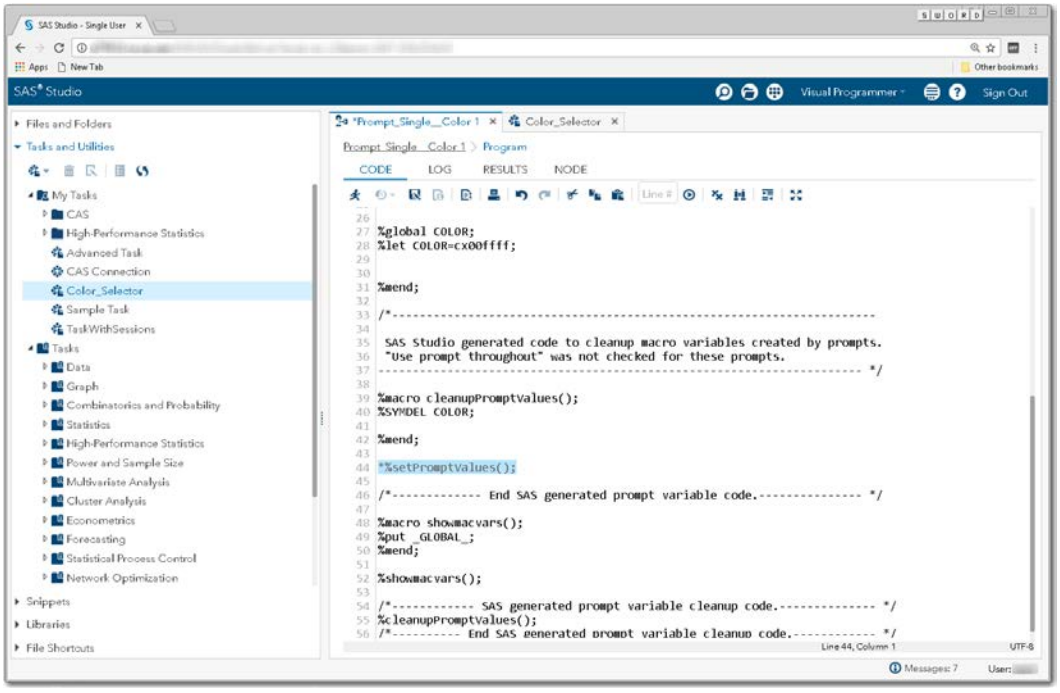
Display 270 - Replacement Task for Color Prompt

2. Save the replacement task to your **My Tasks** folder.
3. Drag the task from **My Tasks** into your converted process flow.
4. Link the output port of the Color_Selector node to the input port for the converted Program node.



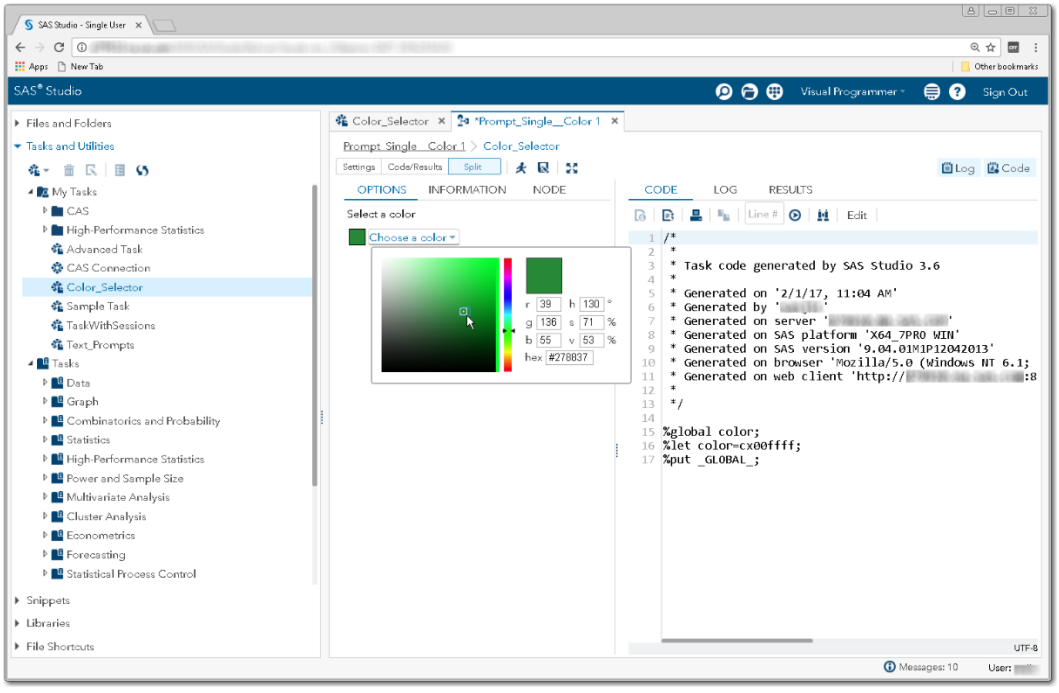
Display 271 - Color_Selector Task Linked to Program Node

5. Comment out the `%setPromptValues()` macro call from the converted Program node. The macro code generated by the Color_Selector task replaces this code.



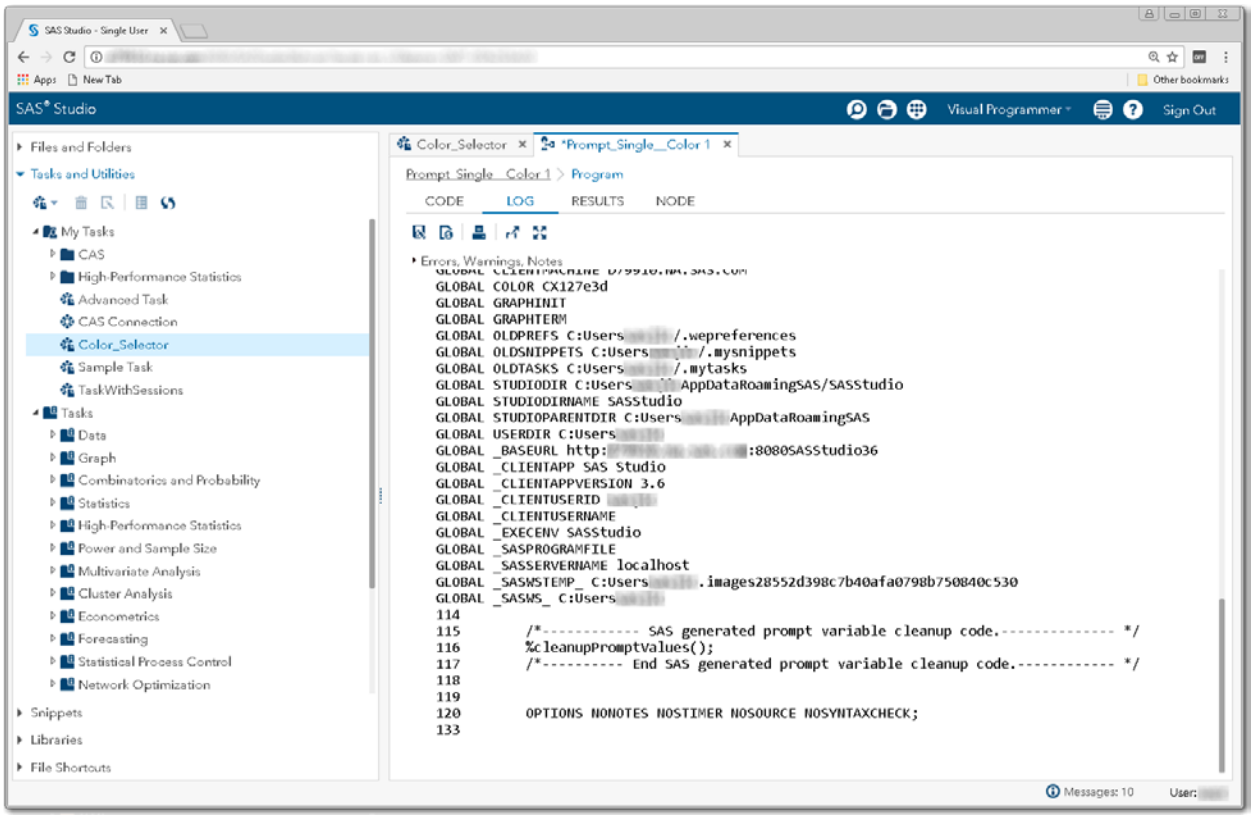
Display 272 - Commented Out %setPromptValues Macro Call

To run your flow with a different color value than the default value, open the Color_Selector node and select a color.



Display 273 – User Interface and Generated Code for Color_Selector Task

When you run the process flow, the global Color variable is set to the value of the selected color.

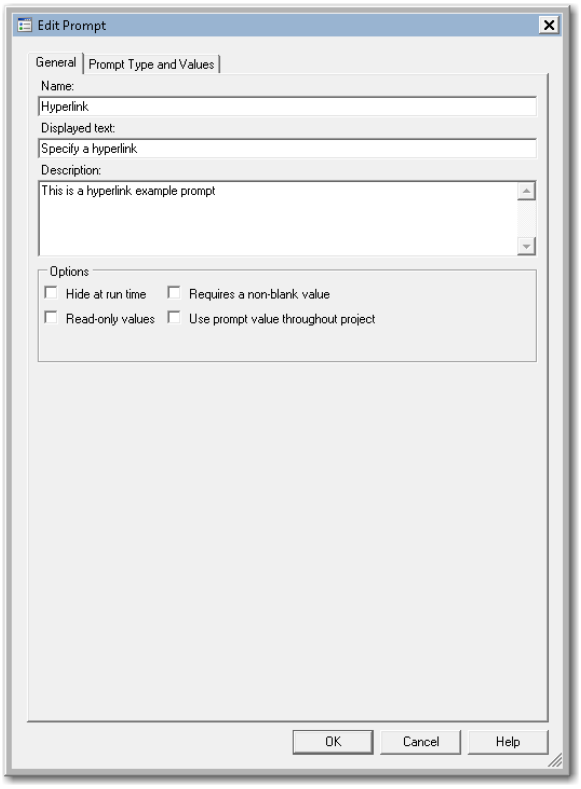


Display 274 - Macro Variable With Updated Value for Color_Selector Task

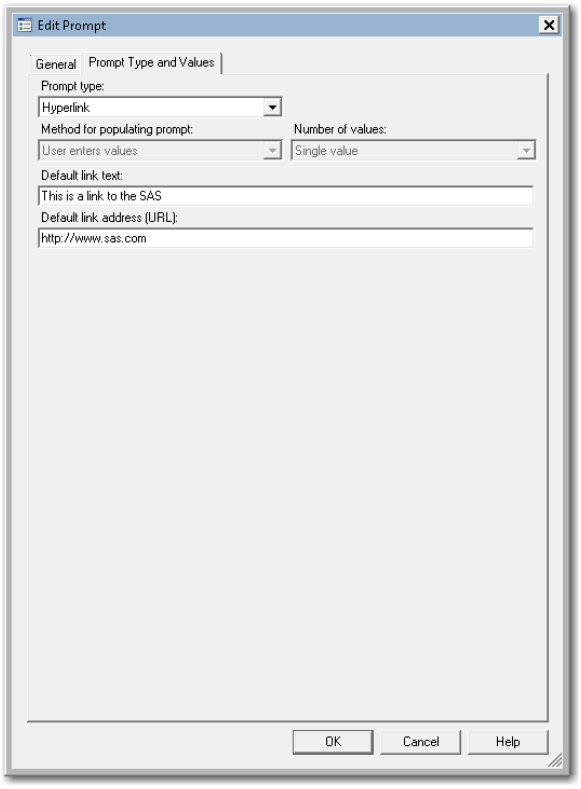
Hyperlink

SAS Enterprise Guide

In this example, a hyperlink prompt named Hyperlink is defined as shown in the following two displays.

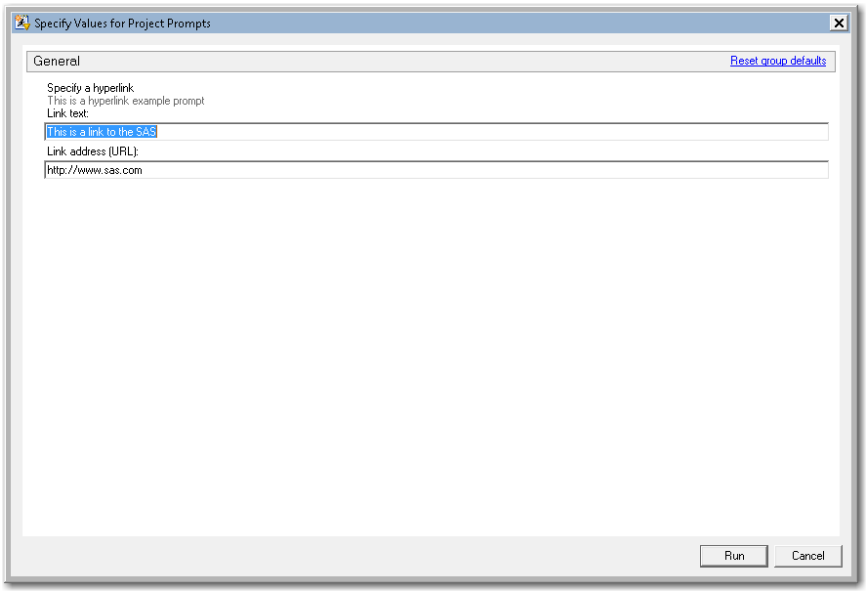


Display 275 - General Properties for Hyperlink Prompt



Display 276 - Type and Values for Hyperlink Prompt

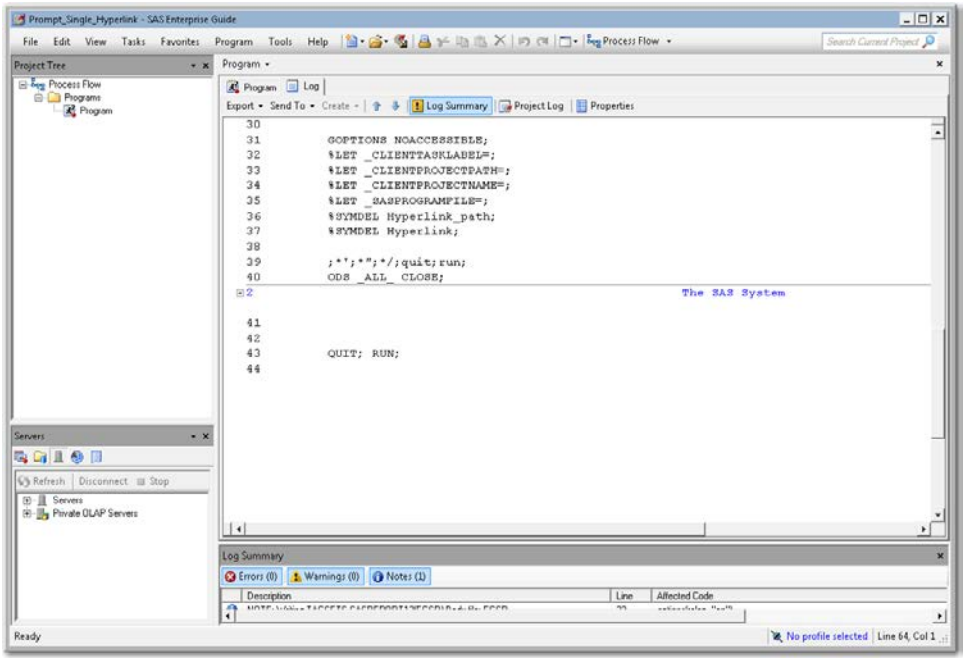
When the code for the prompt is run, the following dialog box appears.



Display 277 - Hyperlink Prompt in Prompt Dialog Box

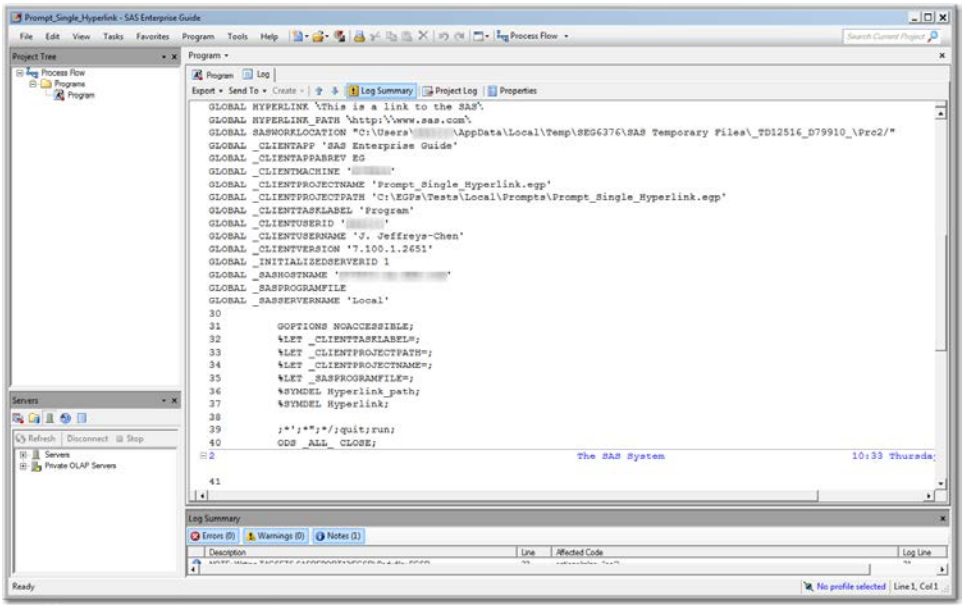
If the user leaves the default values in the hyperlink prompt field, the following code is generated by SAS Enterprise Guide in the Program node that depends on the prompt.

The %LET statements assign the values specified in the prompt dialog box to the Hyperlink* macro variables.



Display 278 - %LET Statements for Hyperlink Prompt

Because the **Use prompt value throughout project** option is not checked for this prompt, the %SYMDEL statements remove the macro variables at the end of the program.

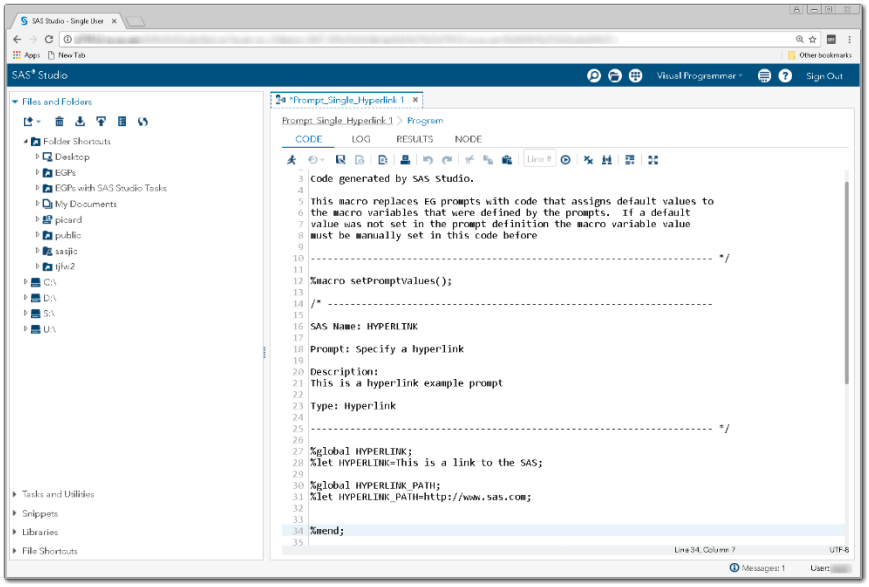


Display 279 - Hyperlink Macro Variable GLOBAL definitions

SAS Studio

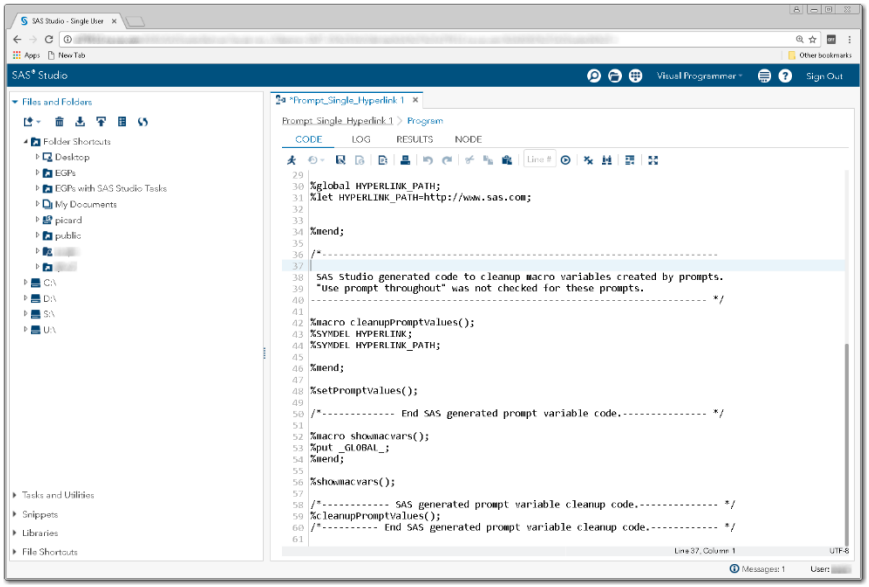
The following display shows code that is added to the converted Program node for the hyperlink prompt in SAS Enterprise Guide.

Global variables named HYPERLINK and HYPERLINK_PATH are created. The %LET statements assign the default values to HYPERLINK and HYPERLINK_PATH. If you want to run your process flow using a different input value for the HYPERLINK prompt, you must manually update the values of the macro variables in the %LET statement.



Display 280 - Macro Variables for Hyperlink Prompt

Because the **Use prompt value throughout project** option is not checked, the %SYMDEL statements remove the HYPERLINK macro variables.



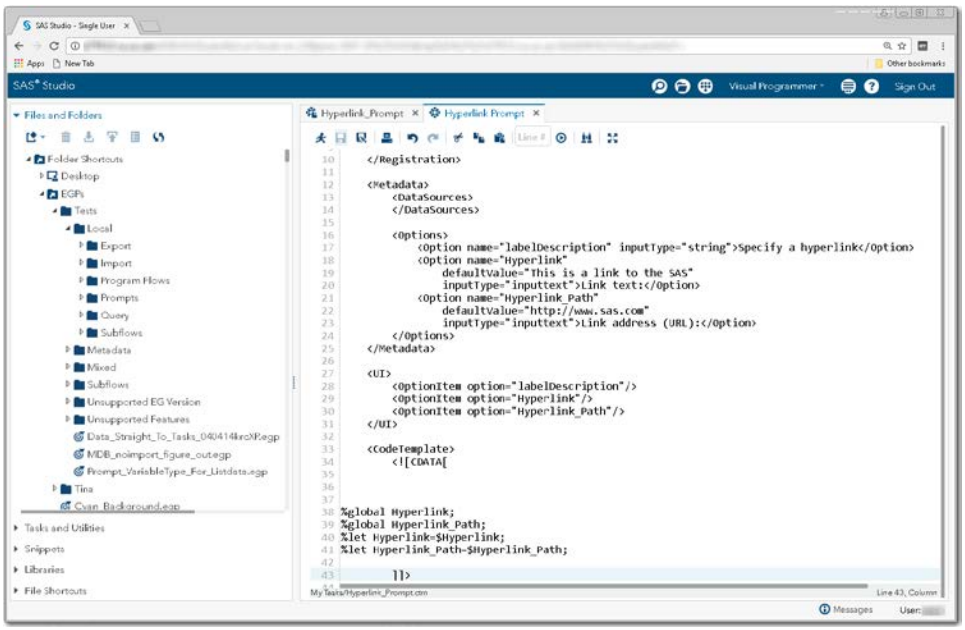
Display 281 - %SYMDEL Statements Remove the HYPERLINK Macro Variables

Substituting a SAS Studio Task for a Hyperlink Prompt

1. Create a SAS Studio task with a control that replaces the hyperlink prompt.
    - Add a label and a text input control for the hyperlink and the hyperlink path. For the name of the input control, use Hyperlink and Hyperlink_Path.
    - Set the default value to the default value shown in the generated `setPromptValues()` macro in the converted Program node.
    - Change the strings of the input controls to match the strings in the hyperlink prompt.

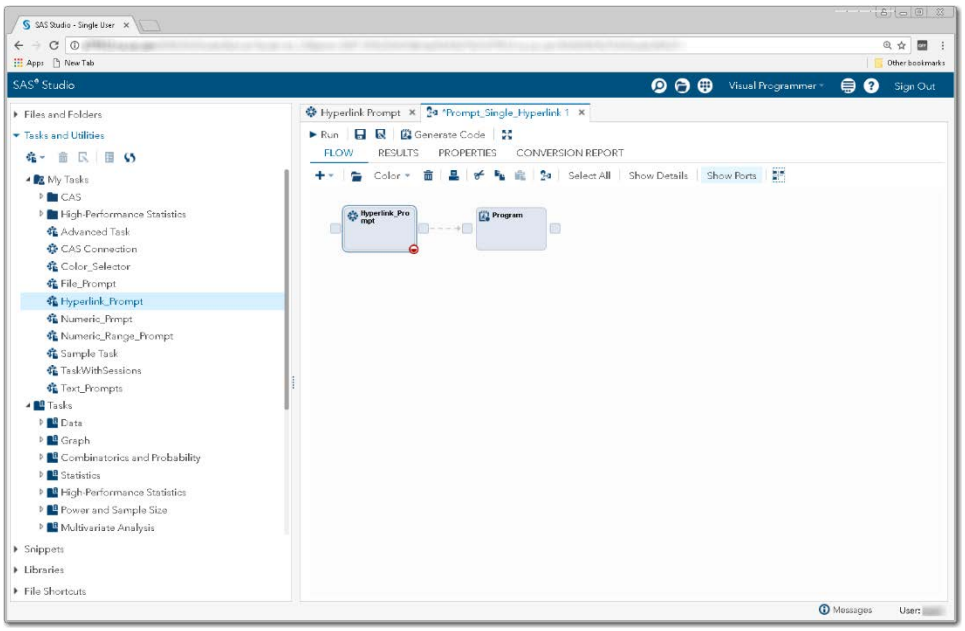
The following code is an example of a task that could be used as the hyperlink prompt.

```
<?xml version="1.0" encoding="UTF-8"?>
<Task schemaVersion="5.1" runNLS="never">
       <Registration>
       <Name>Hyperlink Prompt</Name>
       <Description>This is a hyperlink prompt task.</Description>
       <GUID></GUID>
       <Procedures></Procedures>
       <Version>3.6</Version>
       <Links></Links>
       </Registration>

       <Metadata>
              <DataSources>
              </DataSources>

              <Options>

                     <Option name="labelDescription" inputType="string">
                         Specify a hyperlink
                     </Option>

                     <Option name="Hyperlink"
                             defaultValue="This is a link to the SAS"
                             inputType="inputtext">
                         Link text:
                     </Option>

                     <Option name="Hyperlink_Path"
                             defaultValue="http://www.sas.com"
                             inputType="inputtext">
                          Link address (URL):
                     </Option>

              </Options>
       </Metadata>

       <UI>
              <OptionItem option="labelDescription"/>
              <OptionItem option="Hyperlink"/>
              <OptionItem option="Hyperlink_Path"/>
       </UI>

       <CodeTemplate>
              <![CDATA[

%global Hyperlink;
%global Hyperlink_Path;
%let Hyperlink=$Hyperlink;
%let Hyperlink_Path=$Hyperlink_Path;

              ]]>
       </CodeTemplate>
</Task>
```



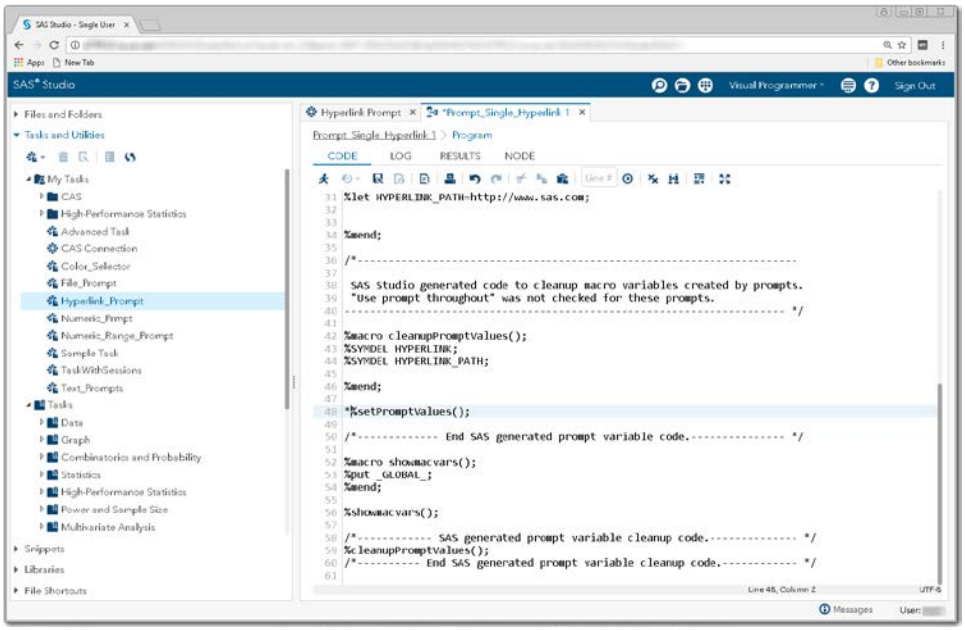
Display 282 - Replacement Task for Hyperlink Prompt

2. Save the new task to your **My Tasks** folder.
3. Drag the task from **My Tasks** into your converted process flow.
4. Link the output port from the Hyperlink_Prompt task to the input node for the converted Program node.



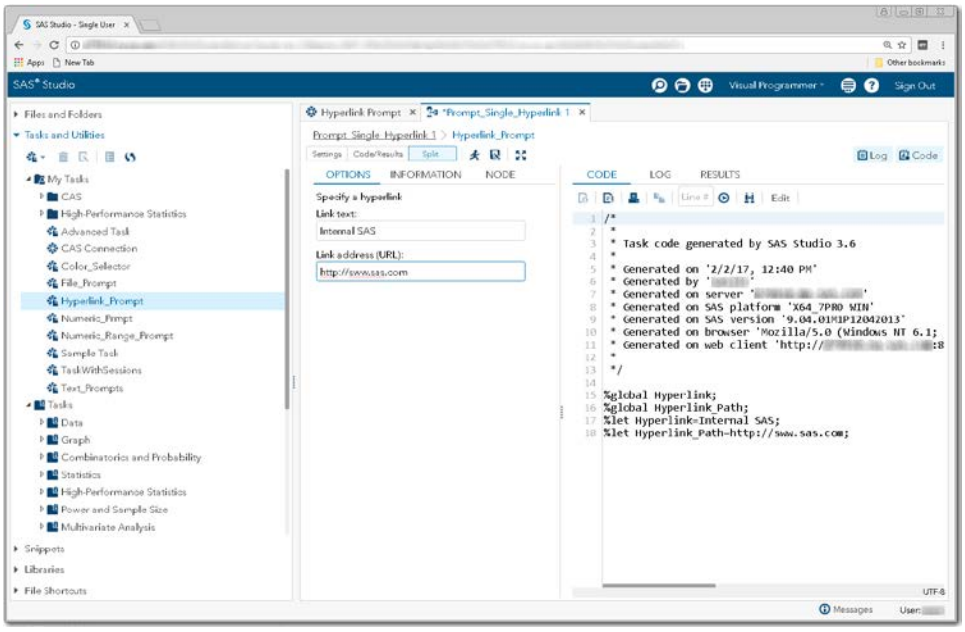
Display 283 - Hyperlink_Prompt Task Linked to Program

5. Comment out the `%setPromptValues()` macro call from the converted Program node. The macro code generated by the hyperlink task replaces this code.



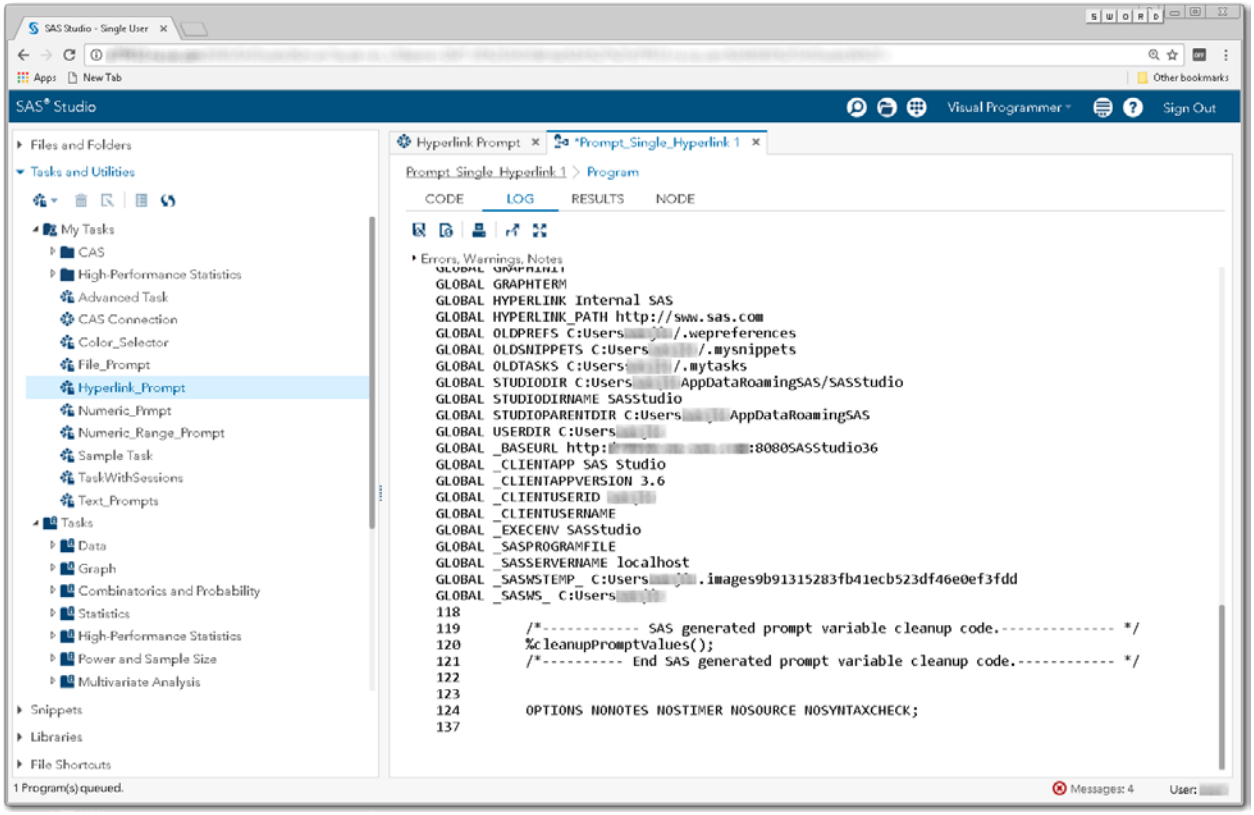
Display 284 - Commented Out %setPromptValues Macro Call

To run your flow with different Hyperlink values than the default values, open the Hyperlink_Prompt node and specify different values.



Display 285 – User Interface and Generated SAS Code for Hyperlink_Prompt Task

When you run the process flow, the global Hyperlink* variables are set to the values specified in the task.



Display 286 – Macro Variables for Hyperlink_Prompt Task with Updated Values

MULTIPLE FLOWS

If there are multiple process flows defined in a SAS Enterprise Guide project, they will appear as subflows when the EGP is converted to a SAS Studio process flow.

EGP ELEMENTS THAT ARE NOT SUPPORTED BY SAS STUDIO

Many elements of SAS Enterprise Guide projects are not currently supported by SAS Studio.

- project tree
- submit to grid
- Decision Manager elements
- scheduling
- Git source control
- explore data list
- process flow zoom – Note: In SAS Studio, you can use browser zoom instead.
- process flow grid view
- process flow layout

SAS ENTERPRISE GUIDE MACROS

Code generated by the SAS Enterprise Guide tasks sometimes includes references to macros. The following macros are also included in SAS Studio:

- _eg_conditional_dropds - Used by code generated for EGP Query nodes.

MACRO VARIABLES IN SAS ENTERPRISE GUIDE

AVAILABLE IN SAS STUDIO

- _SASSERVERNAME
- _CLIENTUSERID
- _CLIENTUSERNAME
- _CLIENTMACHINE

NOT AVAILABLE IN SAS STUDIO

Because these SAS Enterprise Guide macros variables are not set in SAS Studio, you see warnings if your code uses them.

- _CLIENTAPP
- _CLIENTPROJECTNAME
- _CLIENTPROJECTPATH
- _CLIENTTASKFILTER
- _CLIENTTASKLABEL
- _CLIENTUSERNAME
- _CLIENTVERSION
- _SASHOSTNAME

- _SASPROGRAMFILE

LIMITATIONS

- Only EGP files saved with SAS Enterprise Guide 7.1 or later are supported.

Interesting behaviors:

- When the process flows are opened the first time in SAS Studio, they are "Arranged" using the same functionality that you get when you press the "Arrange" button on the SAS Studio process flow.

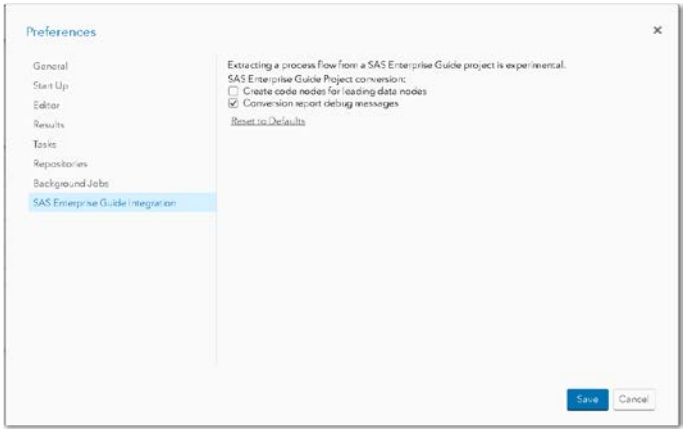
FUTURES

- Data node support in SAS Studio process flows
- Reformatted conversion report
- Command line interface
- Prompt support in SAS Studio process flows

DEBUGGING

To find the unpacked EGP directory, select the **Conversion report debug messages** option in SAS Studio. This option is available from the Preferences window.

Note: The **Create code node for leading data nodes** option is no longer supported (with the 3.6 hotfix) and will be removed in a future release. In SAS Studio 3.6 (with the 3.6 hotfix), all Data nodes are represented in the converted process flow.



Display 287 - SAS Enterprise Guide Integration Preferences

ENABLING EGP EXTRACTION FEATURE

To allow the opening of EGP files in SAS Studio, the SAS Studio Administrator can sets the `webdms.allowEGPOpen` property to `true` in the config.properties file:

```
webdms.allowEGPOpen=true
```

FREQUENTLY ASKED QUESTIONS

Q. Can I save my converted SAS Studio process flows back to SAS Enterprise Guide projects?

A. No.

REFERENCES

SAS Institute Inc. 2016. *SAS Studio 3.6: User's Guide*. Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2016. *SAS Studio 3.6: Developer's Guide to Writing Custom Tasks*. Cary, NC: SAS Institute Inc.

CONCLUSION

The new EGP conversion functionality is just beginning to take shape. Process flows from SAS Enterprise Guide can be opened in SAS Studio. Process flow nodes are transformed to nodes that SAS Studio supports, but these nodes might not function exactly the same way they do in SAS Enterprise Guide. Currently, SAS Studio process flows cannot opened in SAS Enterprise Guide.

CONTACT INFORMATION

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

Jennifer Jeffreys-Chen
SAS Institute Inc.
919-531-6208
J.Jeffreys-Chen@sas.com

To contact your local SAS office, please visit: sas.com/offices