Paper AD02-2011

Take a Fresh Look at SAS® Enterprise Guide®: From point-and-click ad hocs to robust enterprise solutions

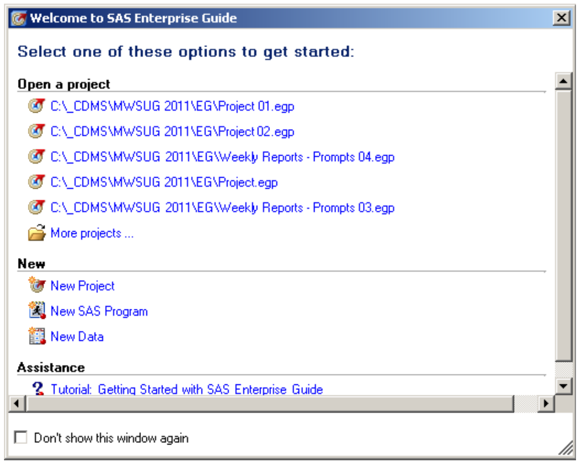
Chris Schacherer, Clinical Data Management Systems, LLC

ABSTRACT

Early versions of SAS Enterprise Guide (EG) met with only lukewarm acceptance among many SAS programmers. As EG has matured, however, it has proven to be a powerful tool not only for end-users less familiar with SAS programming constructs, but also for experienced SAS programmers performing complex ad hoc analyses and building enterprise class solutions. Still, many experienced SAS programmers fail to add EG to their SAS toolkit. They face the barriers of an unfamiliar interface, new nomenclature, and uncertainty that the benefits of using EG outweigh the time spent mastering it. Especially for this group, (but also for analysts new to SAS), the present work attempts to orient new EG users to the interface and nomenclature while teaching them how to achieve common data management and analytic tasks they perform with ease in SAS. In addition, EG concepts and techniques that focus on using EG as a development environment for producing end-user analytic solutions are described.

INTRODUCTION

The first reaction many SAS users have to Enterprise Guide (EG) is "I don't need a point-and-click interface; I'm a real SAS programmer". The idea of clicking on an interface widget to perform a PROC SORT, for example, (instead of simply typing "PROC SORT DATA=….") seems like a frivolous piece of functionality. Others, having further considered why SAS would develop a tool like Enterprise Guide, might even become worried that their livelihoods are being threatened—thinking "if people who do not know SAS can perform these functions on their own, why does my organization need me?" Still others see the flow-diagram-inspired "Process Flows" and imagine how much faster they might crank out the endless stream of ad hoc queries with which they are bombarded once they no longer have to search for (and type) all of those cryptic database table and variable names. "Better yet", this group imagines, "perhaps I could empower my users to do some of this work themselves and I could focus on creating more advanced data management and analytic tools for my organization". SAS Enterprise Guide has something to offer users from each of these perspectives.


Welcome to SAS Enterprise Guide
Select one of these options to get started:
Open a project
C:_CDMS\MWSUG 2011\EG\Project 01.egp
C:_CDMS\MWSUG 2011\EG\Project 02.egp
C:_CDMS\MWSUG 2011\EG\Weekly Reports - Prompts 04.egp
C:_CDMS\MWSUG 2011\EG\Project.egp
C:_CDMS\MWSUG 2011\EG\Weekly Reports - Prompts 03.egp
More projects ...
New
New Project
New SAS Program
New Data
Assistance
Tutorial: Getting Started with SAS Enterprise Guide
Don't show this window again

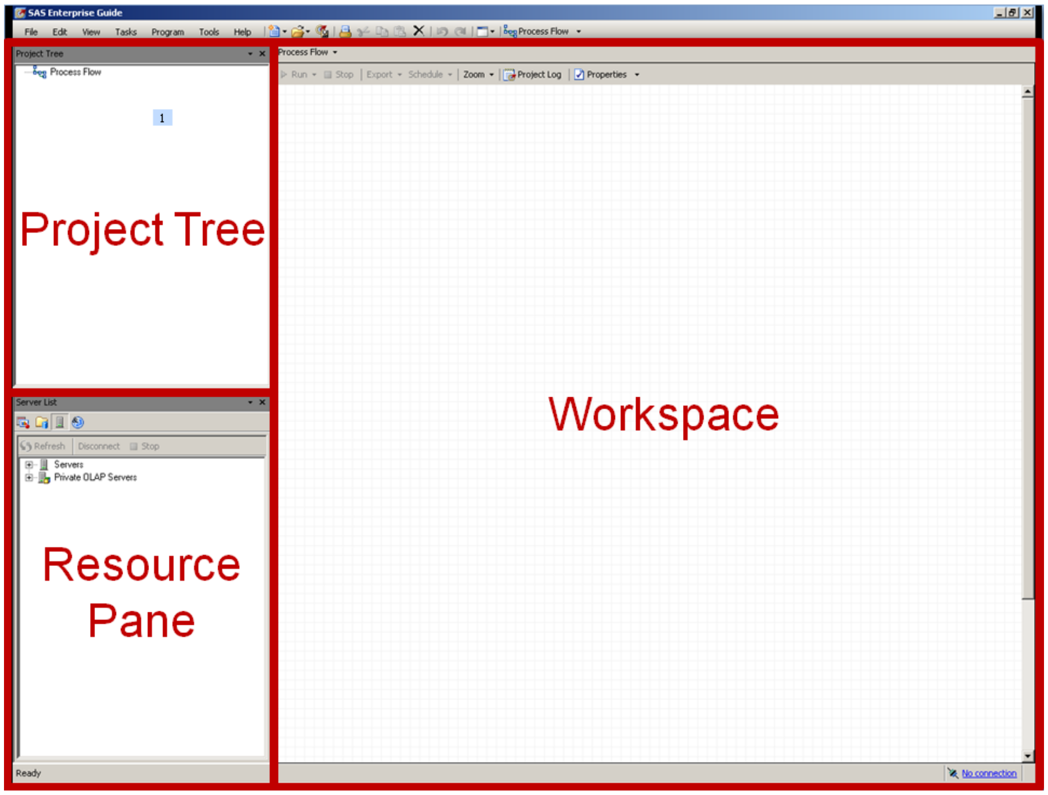
Once a SAS programmer is introduced to EG and learns both the available functionality and the limitations of "point-and-click programming" he or she discovers a programming environment that both (a) empowers a broader community of end-users to transform data into information and (b) provides new opportunities to apply their SAS knowledge to the development of more elegant data management and analytic tools. The problem for these programmers, however, is that when they sit down to create their first EG "program", they often realize "I have no idea how this thing works." In fact, the splash screen with which they are presented at startup poses a question for which they are not entirely prepared—do you want to start a new project or open an existing one? The one option that seems manageable from this splash screen is "New SAS Program". After all, you know what a SAS program is; you have written hundreds or thousands of them over your career. But creating a new SAS program within EG (though a very meaningful part of creating an EG Project), just delays the inevitable, you need to understand the interface and nomenclature used to organize your work within Enterprise Guide.

In order to start the transition to EG, it is important to start adapting to a new nomenclature. This transition begins with understanding that a **Project** in Enterprise Guide "is a collection of related data, tasks, results, programs, and notes" (Slaughter & Delwiche, 2010). You might wonder whether a Project is just the same as a SAS Program; after all, a SAS program is little more than a collection of related LIBNAME statements, comments, PROC and DATA steps, and the results of those PROC and DATA steps. Further, you can %INCLUDE SAS programs within one another, so even the idea of having multiple, related programs referenced within a single driver program (Fecht, 2009) is not new to the SAS programmer. So, in what ways does a Project differ from a Program? First, not only can

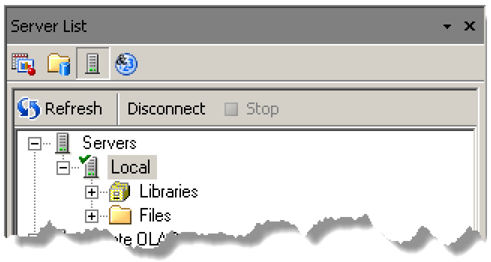
EG Projects include links to an external SAS Program, but, as is demonstrated later in the paper, can contain **Embedded SAS Programs** which exist solely within the confines of the project—having no external .SAS file. Further, EG contains built-in facility for **Conditional Processing** to control the branching of code execution and user interface components for presenting users with **Prompts** for selection or assignment of values that will drive code execution—for example, by the assignment of macro variable values in a PROC SQL WHERE clause. Beyond these differences, the goal of the Enterprise Guide Project and the SAS Program are the same—manipulation and transformation of data for reporting and analysis. The main difference between the two software packages is that the SAS System accomplishes these tasks strictly through the execution of user-written code, whereas Enterprise Guide focuses on presenting the user with graphical user interface (GUI) components that gather the specifications for SAS code that is then generated by the software. The first step in learning how to create these point-and-click Projects is to gain a basic understanding of the user interface in which they are built.

ENTERPRISE GUIDE INTERFACE

After launching Enterprise Guide and choosing "New Project" to navigate past the splash-screen[1], you are presented with a screen comprised of two docked windows (Resources and Project Tree) and the Workspace—which technically is not a "window" since it cannot be closed, minimized, etc. independently from the application.



RESOURCES WINDOW. The Resources window is located in the lower left corner of the EG screen. This window, as its name suggests, organizes the resources available to the user. The four main types of resources available to you are: Servers, SAS Folders, Tasks, and Prompts.



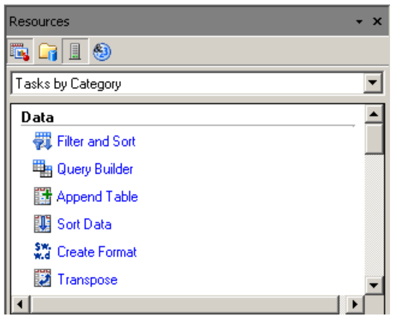
SERVERS refers to SAS Servers to which you have access. If you are running EG in stand-alone mode (i.e., not connected to a Workspace, Stored Process, or other remote server), by default EG will attempt to connect to a "Local" server that represents your local installation of the SAS System. The fact that SAS is connecting to a SAS Server (either your local installation or another Server installation) belies the key characteristic describing EG; it is essentially a graphical user interface (GUI) to the SAS System. To confirm that EG is using your local SAS installation, Start EG and navigate to the SAS Server "Local". After

[1]Of course, if at some point you checked "Do not show this window again", you will no longer see the splash-screen at start-up, but will instead be taken directly to a new project.

"Local" is started, launch SAS. You should notice that SAS gives you a warning that "User Preferences will not be saved". This is because your local SAS environment is already in use—as the SAS engine for Enterprise Guide. Enterprise Guide generates SAS code necessary to execute the Tasks you have specified and uses the local SAS engine to run that code.

In addition to LOCAL, you can connect to other SAS Servers (e.g., MetaData, Workspace, etc.) that give you access to data sources, macros, and stored processes available throughout your organization's SAS infrastructure, but the scope of the current paper will be limited to the LOCAL server.

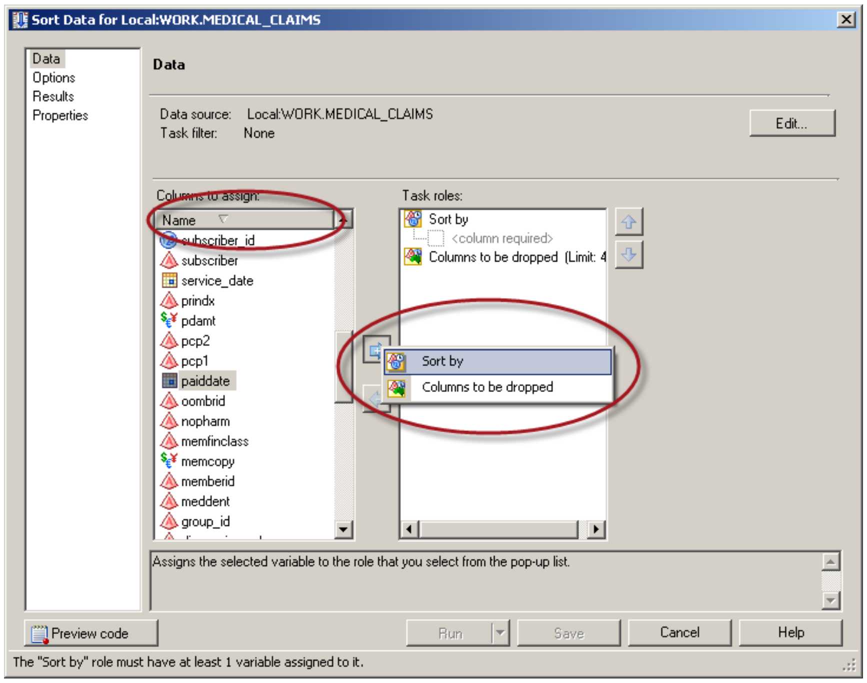
SAS FOLDERS SAS Folders are directories defined in a SAS Metadata Server to standardize access to commonly used repositories throughout your organization. As they expand beyond the scope of the LOCAL server, they will not be discussed further here—but see SAS (2011) for further information about this resource.



TASKS A great deal of the work performed in an EG Project is specified by adding **Tasks** to the project. As seen in the figure to the right, some of the entries in the TASK resource have names that are remarkably similar the names of PROCS you might use in SAS. Many of these tasks map directly to a SAS PROC; for example, using the "Sort Data" task sorts a dataset in the Project just as PROC SORT would in a SAS program. Similarly, just as PROCs have options and keywords that drive "how" the PROC will be executed, so do Tasks. Each task has its own dialog window and/or wizard that allows the user to specify the options to apply to how the task is executed. This is where many experienced SAS programmers first throw up their hands in exasperation because they can very likely write this SAS code faster than they can point and click their way through the task.

But suppose you want to sort a dataset containing a large number of cryptically named variables associated with (say) healthcare claims. If you do not use these data frequently and you want to sort the claims by the date on which they were paid, you might only remember that the variable on which you want to sort probably has the word "paid" and "date" in it, but you might not remember whether it is "paiddate", "paid_date", or "date_paid". In SAS you would either right-click on the dataset in the LIBRARY window and choose "View Columns", try running PROC SORT with each derivation of the name until you hit the correct variant, or open the dataset and scroll across the screen looking for the variable of interest.

By contrast, in the Sort Data Task, you can simply click the "Name" heading under "Columns to Assign" to sort the column names in ascending or descending order, scroll down the list and click on the variable of interest—in this case "paiddate". Once found, however, you are confronted with another concept that at first blush may seem foreign—the **Role** that the variable is to play within the task. Intuitively, it is easy to make the leap that if you want to sort the dataset by "paiddate" this variable is to play the "Sort by" role in the current task. As seen in the adjacent figure, once the variable of interested is selected, click the button to select the Role to be played by that variable in the current task. Alternatively, you can drag "paiddate" to the "Sort by" role in the Task roles list.


Sort Data for Local:WORK.MEDICAL_CLAIMS

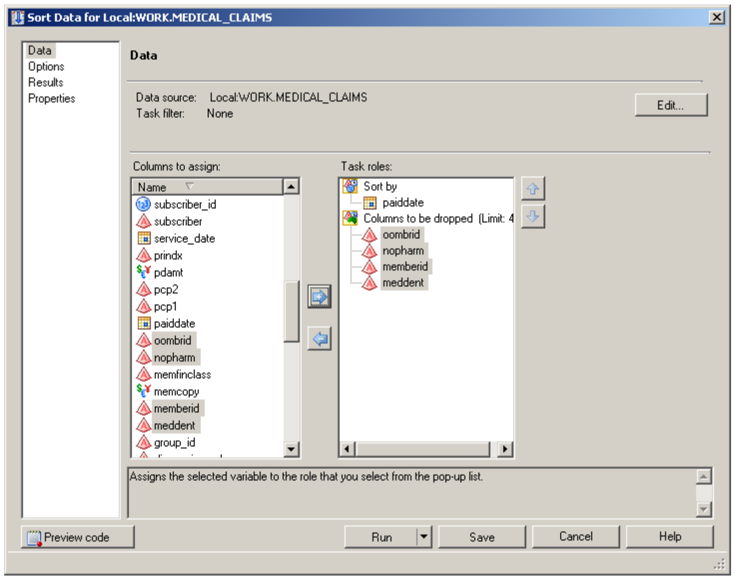
Data source: Local:WORK.MEDICAL_CLAIMS
Task filter: None

Assigns the selected variable to the role that you select from the pop-up list.

The "Sort by" role must have at least 1 variable assigned to it.

The "Sort Data" task also has another useful role, "Columns to be dropped". Together, the use of these two roles results in a very quick and efficient way to both sort a data set by one or more columns and reduce the variables in the dataset. Although in some ways it seems like a new concept, the Task Role is really just an explicit name given to a concept that SAS programmers have come to accepted implicitly after many years of reading syntax examples like the following (SAS, 2009):

```
PROC SORT <collating-sequence-option>
     <other option(s)>;
  BY <DESCENDING> variable-1
     <...<DESCENDING> variable-n>;
```



Data

Data source: Local:WORK.MEDICAL_CLAIMS
Task filter: None

Columns to assign: subscriber_id, subscriber, service_date, prindx, pdamt, pcp2, pcp1, paiddate, oombrid, nopharm, memfinclass, memcopy, memberid, meddent, group_id

Task roles:
- Sort by
  - paiddate
- Columns to be dropped (Limit: 4
  - oombrid
  - nopharm
  - memberid
  - meddent

Assigns the selected variable to the role that you select from the pop-up list.

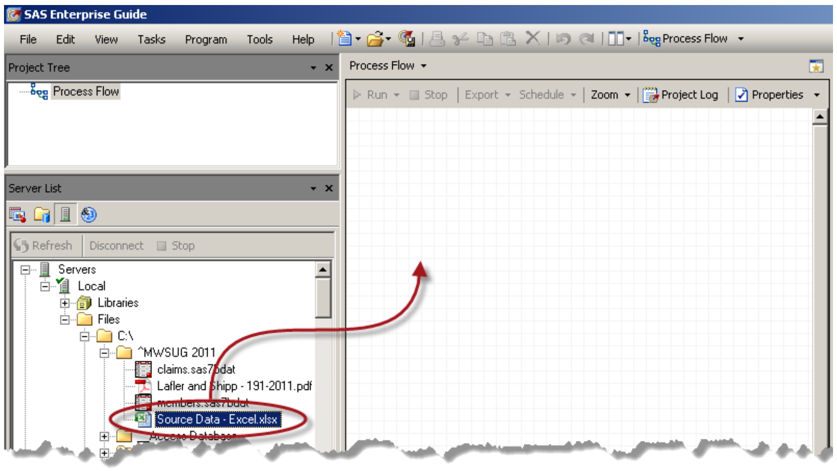
When executed as part of your EG Project, the preceding Sort Data task generates the following PROC SORT code:

```
PROC SORT DATA=WORK.MEDICAL_CLAIMS
     OUT=WORK.SORTSortedMEDICAL_CLAIMS(
           LABEL="Sorted WORK.MEDICAL_CLAIMS"
           DROP= oombrid nopharm memberid meddent);
     BY paiddate;
RUN;
```

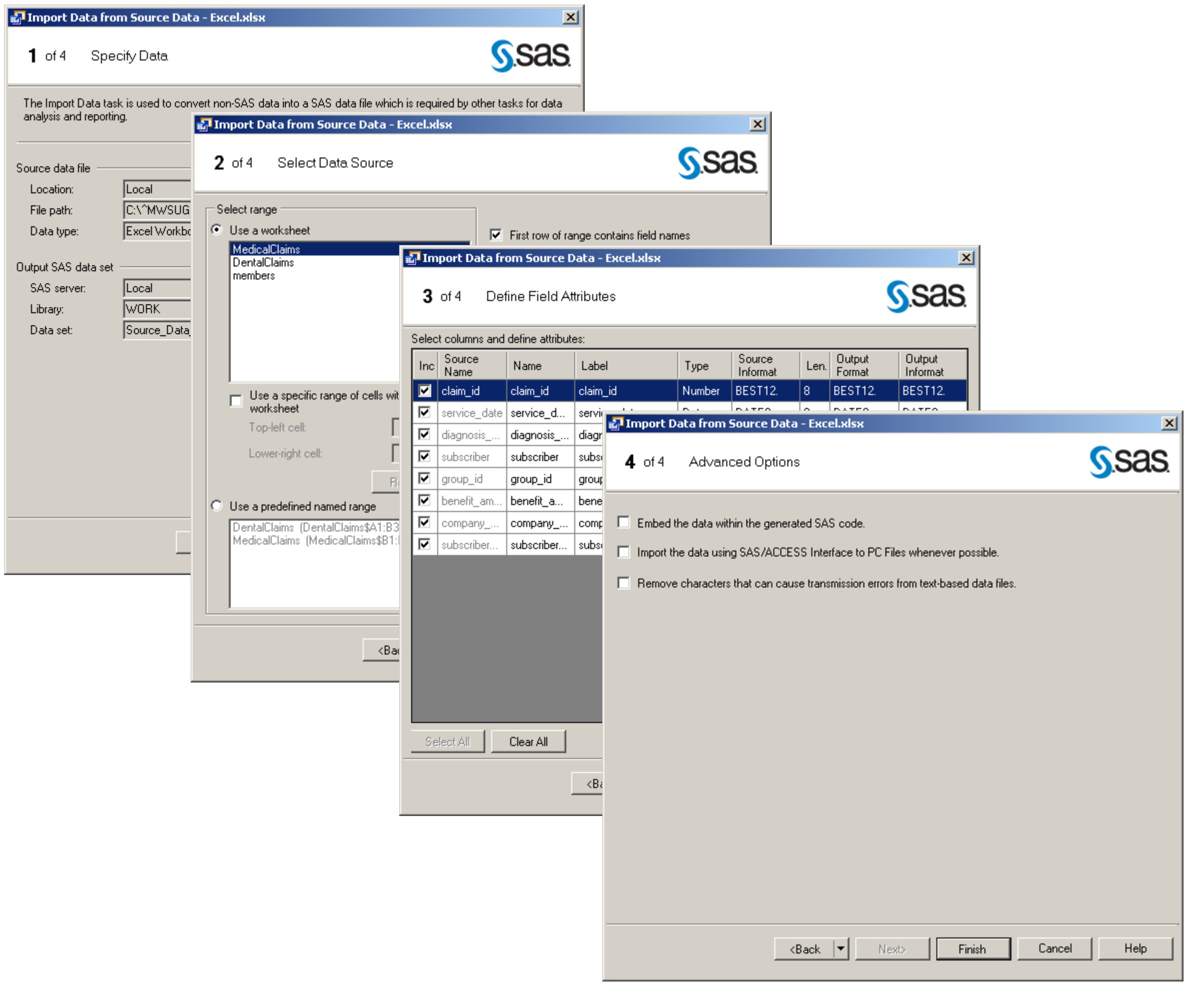
This example demonstrates the critical role Tasks play within EG; they assist the user in generating SAS code by providing an intuitive interface for specifying all of the parts of the syntax that need to be understood by SAS in order to generate and execute code that performs the transformations, analysis, and reports intended by the user. And just as you once did not know PROC SORT from PROC TRANSPOSE, you will have to familiarize yourself with how the different Tasks work. This time around, however, you have the advantage of already knowing how SAS performs these operations. As a result, the time it takes to become proficient with EG is a fraction of the time you struggled through all of the different permutations of the PROCs in your SAS lexicon.

PROMPTS **Prompts** play an important role in creating enterprise-level solutions for analysts and report writers. This resource provides a flexible way to present users with interface components that they can use to indicate single or multiple values that will drive execution of a project. The specified values can be used, for example, to value macro variables used in the WHERE clause of a PROC SQL statement, as the conditional value in an IF/THEN statement in a DATA step, or as a password enabling a database connection. An example of the use of Prompts is provided later in the paper.

WORKSPACE. The workspace is the part of the EG interface that is used to present Process Flows, Document Windows, SAS Code, and Logs. The **Process Flow** in SAS Enterprise Guide is a construct that helps organize the Tasks that are being performed. Although each Project, by default, has at least one Process Flow, you can have multiple Process Flows in a single project—for example, to organize the Tasks comprising "Step 1" and "Step 2" of a complex set of data transformations. In the following example the Excel data file "Source Data – Excel.xlsx" is dragged from the Local server's file structure into the Process Flow.



When this drag-and-drop operation is performed, EG automatically detects that what the user wants to do is IMPORT an external, non-SAS data file into the project, and EG launches the Import Data task. This task walks the user through a four-step wizard to (1) specify the source data file and output dataset, (2) select the Excel worksheet to import, (3) specify data types, informats, and formats for the imported data, and (4) specify any Advanced Options associated with the Task.


Import Data from Source Data - Excel.xlsx

1 of 4 Specify Data

The Import Data task is used to convert non-SAS data into a SAS data file which is required by other tasks for data analysis and reporting.

Source data file
- Location: Local
- File path: C:\"MWSUG
- Data type: Excel Workbo

Output SAS data set
- SAS server: Local
- Library: WORK
- Data set: Source_Data

Import Data from Source Data - Excel.xlsx

2 of 4 Select Data Source

Select range
- (●) Use a worksheet: MedicalClaims / DentalClaims / members
- [✓] First row of range contains field names
- [ ] Use a specific range of cells wit worksheet — Top-left cell: / Lower-right cell:
- ( ) Use a predefined named range: DentalClaims (DentalClaims$A1:B3 / MedicalClaims (MedicalClaims$B1:

Import Data from Source Data - Excel.xlsx

3 of 4 Define Field Attributes

Select columns and define attributes:

Inc	Source Name	Name	Label	Type	Source Informat	Len.	Output Format	Output Informat
☑	claim_id	claim_id	claim_id	Number	BEST12.	8	BEST12.	BEST12.
☑	service_date	service_d...	servi					
☑	diagnosis_...	diagnosis_...	diagr					
☑	subscriber	subscriber	subs					
☑	group_id	group_id	group					
☑	benefit_am...	benefit_a...	bene					
☑	company_...	company_...	comp					
☑	subscriber...	subscriber...	subs					

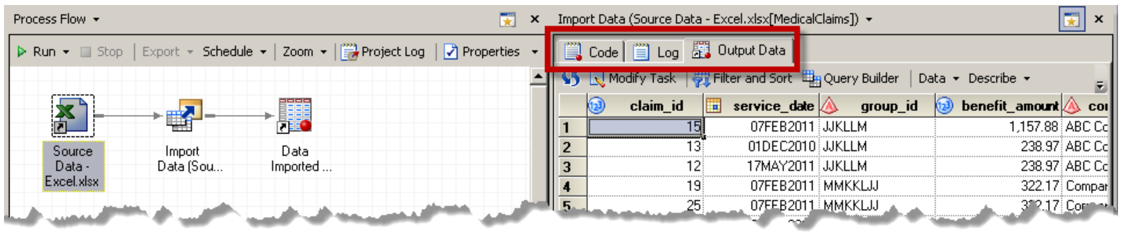
Select All | Clear All

Import Data from Source Data - Excel.xlsx

4 of 4 Advanced Options

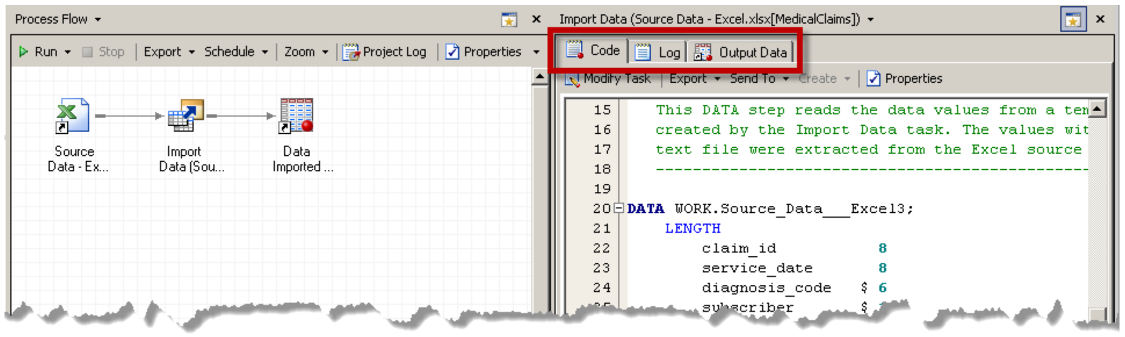
- [ ] Embed the data within the generated SAS code.
- [ ] Import the data using SAS/ACCESS Interface to PC Files whenever possible.
- [ ] Remove characters that can cause transmission errors from text-based data files.

Once all of the steps involved in this Task are specified, clicking "Finish" generates the depiction of the relationship between the source dataset, the import task, and the output dataset in the Process Flow in the left side of the Workspace and the output dataset is rendered in the right side of the Workspace—in a **Document Window**.

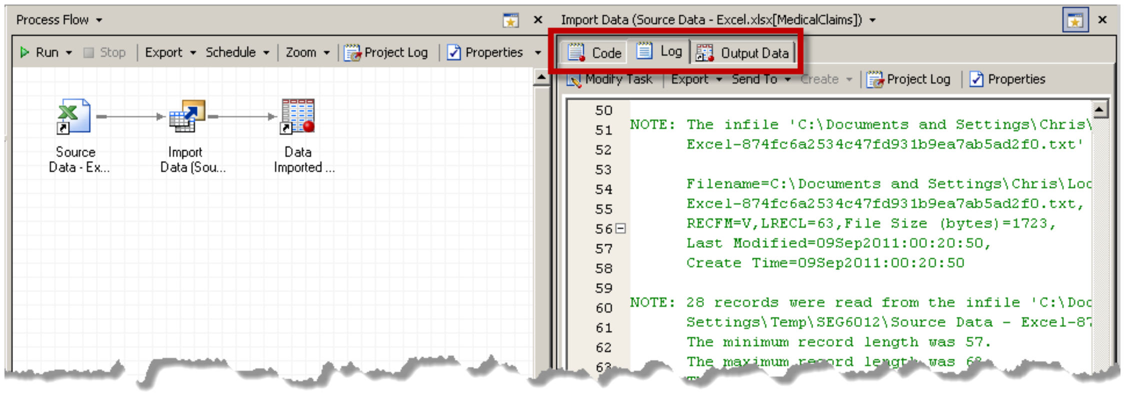


	claim_id	service_date	group_id	benefit_amount	co
1	15	07FEB2011	JJKLLM	1,157.88	ABC Cc
2	13	01DEC2010	JJKLLM	238.97	ABC Cc
3	12	17MAY2011	JJKLLM	238.97	ABC Cc
4	19	07FEB2011	MMKKLJJ	322.17	Compar
5	25	07FEB2011	MMKKLJJ	[illegible]	[illegible]

Note that the Document Window also provides tabs for the SAS code generated by EG as well as the Log entries generated by the execution of that code.



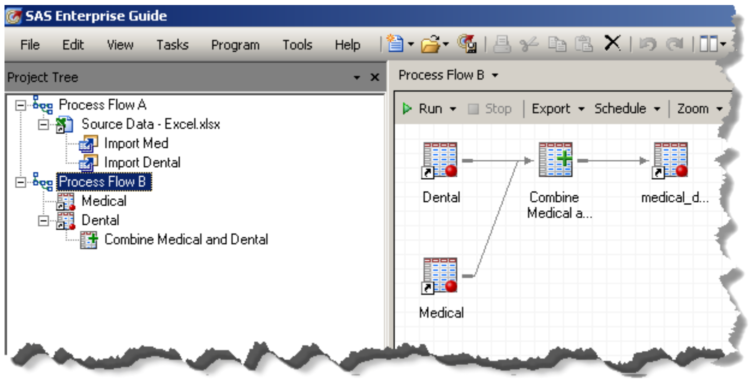
```
   This DATA step reads the data values from a tem
   created by the Import Data task. The values wit
   text file were extracted from the Excel source
   -----------------------------------------------

DATA WORK.Source_Data___Excel3;
    LENGTH
        claim_id              8
        service_date          8
        diagnosis_code      $ 6
        subscriber          $
```



```

NOTE: The infile 'C:\Documents and Settings\Chris\
      Excel-874fc6a2534c47fd931b9ea7ab5ad2f0.txt'

      Filename=C:\Documents and Settings\Chris\Loc
      Excel-874fc6a2534c47fd931b9ea7ab5ad2f0.txt,
      RECFM=V,LRECL=63,File Size (bytes)=1723,
      Last Modified=09Sep2011:00:20:50,
      Create Time=09Sep2011:00:20:50

NOTE: 28 records were read from the infile 'C:\Doc
      Settings\Temp\SEG6012\Source Data - Excel-87
      The minimum record length was 57.
      The maximum record length was 6
```

To summarize the purpose of the Workspace, then, Process Flows are used to organize Tasks, and Document Windows are used to view output datasets that are generated by the Tasks as well as the Code and Log entries associated with execution of those tasks.

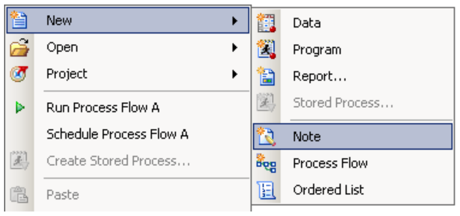
PROJECT TREE. The Project Tree serves to organize the project by providing quick reference to the lineage of datasets, the Tasks that use and generate those datasets, and the Process Flows used to organize the tasks in logically meaningful groupings within the project. In the adjacent Project Tree, we see that the Tasks "Import Med" and "Import Dental" are both part of "Process Flow A". The output datasets "Medical" and "Dental" are used in "Process Flow B" as inputs to the Task "Combine Medical and Dental". Through careful naming of Tasks and datasets and the logical groupings of Tasks within Process Flows, the Project Tree can be a useful tool for quickly navigating through a complex project and for documenting your analytic solution.



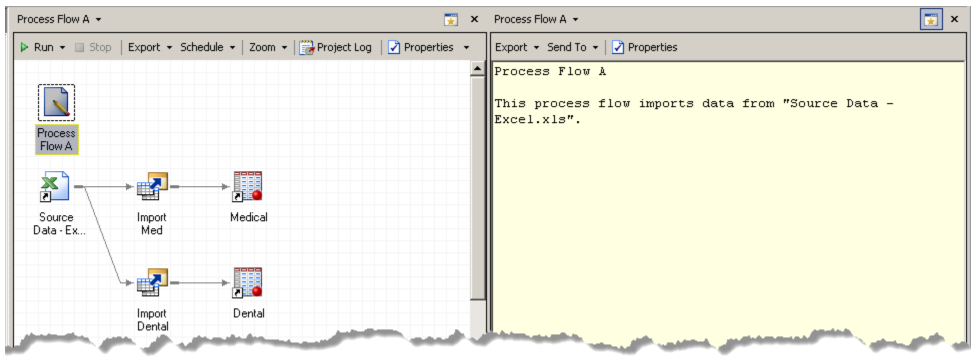
WORKING WITH TASKS AND PROCESS FLOWS

Having been provided with an introductory overview of the EG interface, most SAS Programmers are more than ready to be turned loose and use some **Resources** to build a **Project** comprised of **Tasks** in which dataset variables play specific **Roles**. However, as you should also do in SAS Programs, it is a good idea to first add comments about your Project to each Process Flow.

PROJECT COMMENTING. Although some Enterprise Guide users may have sophisticated source code repository systems, others may rely on file naming conventions and comments within their code to track changes. For this latter group of users, one of the nice features of EG is that you can add **Notes** to your project. To add a Note to a process flow, simply right-click in the workspace and choose "New►Note". A

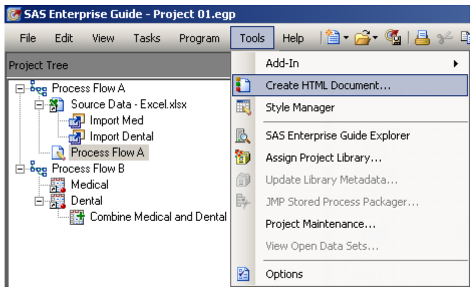


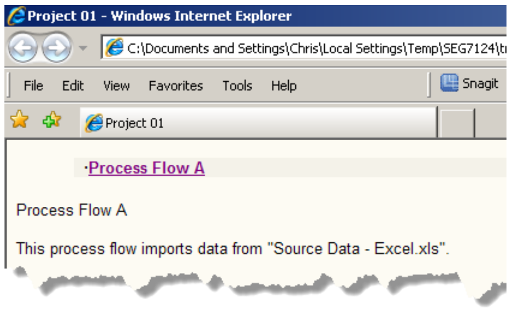
blank Note then opens in the document window, and you can provide the technical information, background, and change history information necessary to effectively maintain the process flow over time.



```
Process Flow A

This process flow imports data from "Source Data -
Excel.xls".
```

The note can then be used to document your project using the "Create HTML Document" option under the "Tools" menu, and generate an HTML document that combines all of the Notes in your Project file.

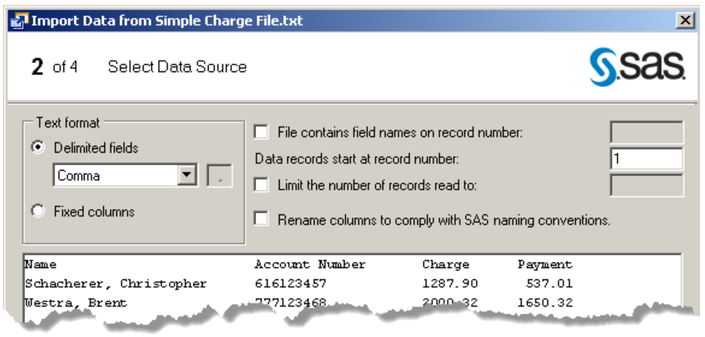



Project 01 - Windows Internet Explorer
C:\Documents and Settings\Chris\Local Settings\Temp\SEG7124\t
File
Edit
View
Favorites
Tools
Help
Snagit
Project 01
·Process Flow A

Process Flow A

This process flow imports data from "Source Data - Excel.xls".

DATA ACCESS. After writing comments to accurately describe the purpose of your project, the next step in creating an EG Project is gaining access to the data with which you need to work. As demonstrated in the earlier example of drag-and-drop data access from the Local server, bringing an Excel file into the Process Flow (by default) triggers the creation of an Import Task. Depending on the file type being imported, you are walked through the import process with import wizards specific to that file type. In contrast to Step 2 in the previous example of importing an Excel file, Step 2 of the Import Task for a .txt file contains the options depicted in the following figure—with radio buttons to specify the format of the records in the file and a drop-down menu to specify the column (variable) delimiter.



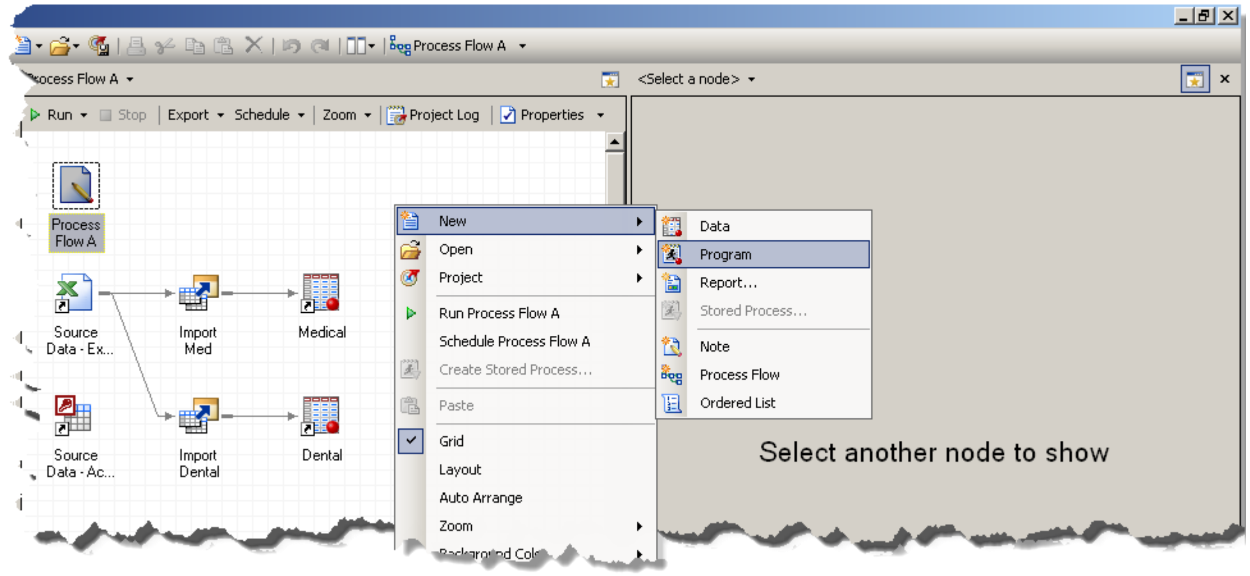
Text format
[✓ Delimited fields] Comma
Fixed columns

File contains field names on record number:
Data records start at record number: 1
Limit the number of records read to:
Rename columns to comply with SAS naming conventions.

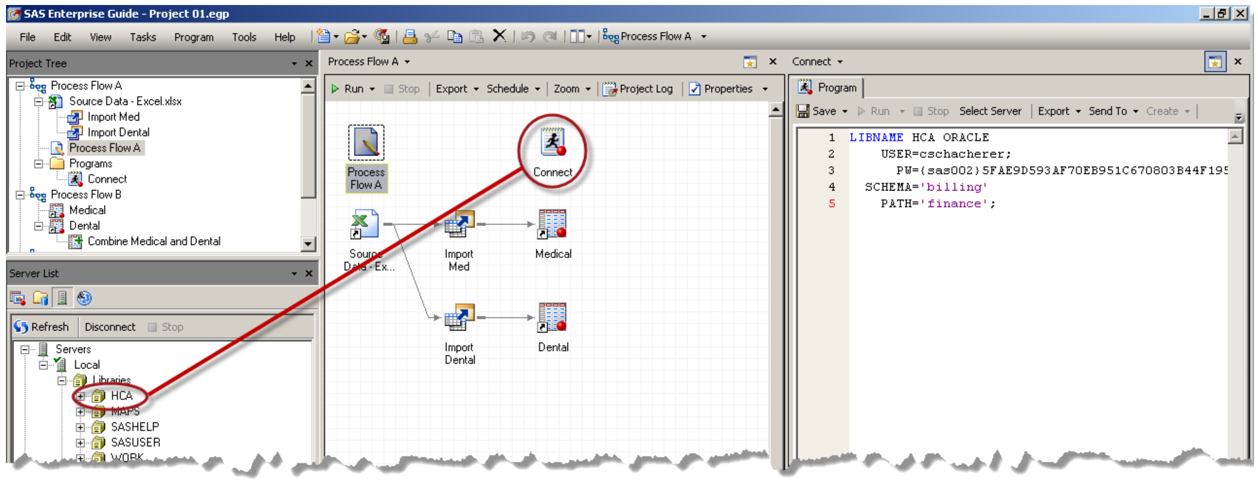
```
Name                    Account Number    Charge    Payment
Schacherer, Christopher 616123457         1287.90    537.01
Westra, Brent           777123468         2000.32   1650.32
```

The ability to drag and drop data files into a Project provides the EG user with a great advantage compared to having to remember the syntax of INFILE statements, PROC IMPORT, and the like for every file type and configuration they might encounter. To be sure, there are files (especially files with multiline records) that are not amenable to this drag and drop approach due to their sophisticated layout, but for the most common file types and layouts this approach provides a very quick and easy way to bring data into your project.

For data that resides in relational databases such as Oracle, SQL Server, DB2, etc., a different approach is required for the Enterprise Guide user accessing the SAS System on his or her local computer. Because the Tasks in Enterprise Guide are built for utilizing SAS to perform analyses and data manipulations (and not for configuring your SAS session), establishing a LIBRARY based on a SAS/ACCESS® connection to a database such as Oracle or DB2 requires some SAS code to be written the old fashioned way—in a SAS Program. To create a SAS Program in your project, simply right-click in the Process Flow and select New ► Program to launch the Enhanced Editor.

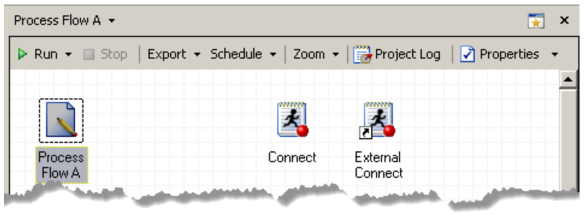


In the following example, a LIBNAME statement is used to define the library "HCA" as a connection to the "billing" database on the Oracle server "Finance". Following execution of this program (renamed "Connect"), a refreshed view of the Local server shows that the project now has access to the HCA library[2]. From this point forward, EG Tasks in the project can access the data tables in the Oracle database—but see, for example, Hemedinger (2007a, 2007b) for an explanation of why you might want to consider pass-through SQL statements when querying large datasets with EG.



```
LIBNAME HCA ORACLE
    USER=cschacherer;
      PW={sas002}5FAE9D593AF70EB951C670803B44F195
  SCHEMA='billing'
    PATH='finance';
```

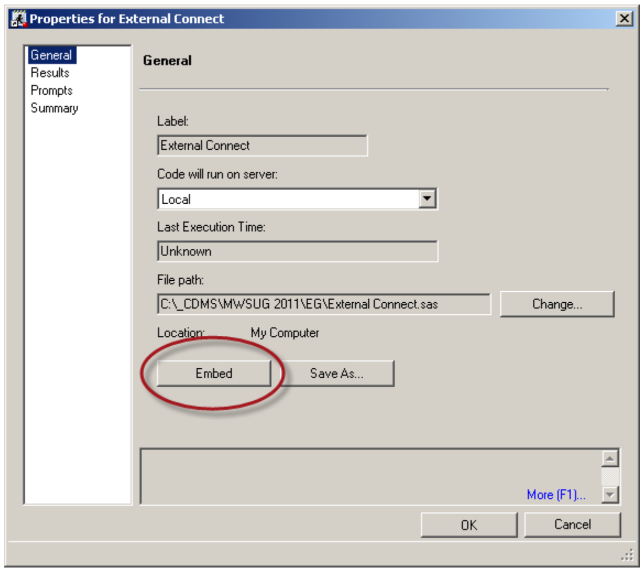
LINKING & EMBEDDING SAS PROGRAMS. In the previous example, the SAS program "Connect" was created within the project as an **Embedded Program**. As such, it exists only within the EG project where it was created; there is no ".sas" file saved externally. If, on the other hand, we had included an existing SAS program in the project by dragging the program from the Files resource on our SAS server, the program would be **Linked** to the project. In the latter case, changes to the program file (which exists outside of the project) would impact the project because when the project is next run following those program changes, the changes are naturally reflected in the "linked" program by virtue of the



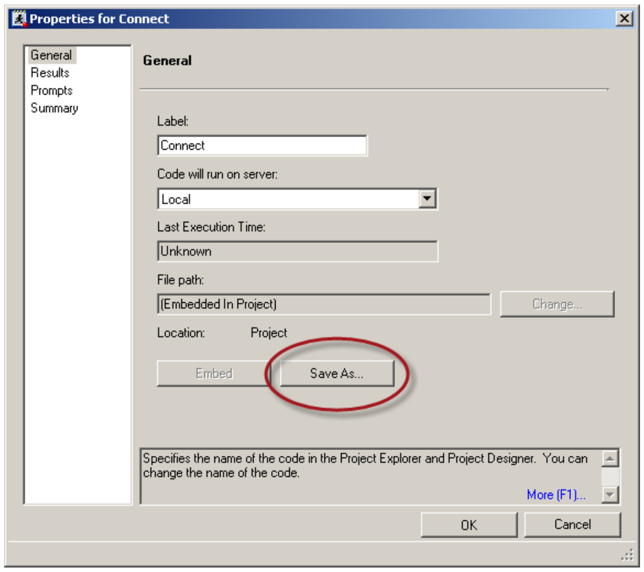
[2] For more information on defining SAS Libraries as connections to databases via SAS/Access see Levine (2001), Schacherer & Westra (2010), SAS (2004a, 2004b).

fact that EG is simply running the external SAS program. In the case of the Embedded program, the only way changes can be made to the program is to open the EG project where the program is saved, and edit the program. The embedded program "Connect" is represented in the flow diagram by the familiar SAS Program icon whereas the linked program "External Connect" is represented by a shortcut icon—denoting, again, that this program exists as a stand-alone SAS program outside of the EG Project.

You might decide later that an embedded program like "Connect" might be useful in several of your EG programs because you often connect to the same database. In that case, you might want to convert it to be an external SAS program that you can link to your projects and manage as a single program (instead of updating embedded programs in multiple projects.) Conversely, you might decide that a program you wrote outside of EG is so highly specialized that it does belong as an integral part of a single EG project. In either case, it is easy to make the desired change. To embed a linked SAS program, simply right-click on the program, choose "Properties", and click on "Embed". Conversely, to save an external copy of an embedded program, simply right-click on the program and choose "Save As" and you will be prompted to specify a name for the saved program and a location in which to save it.



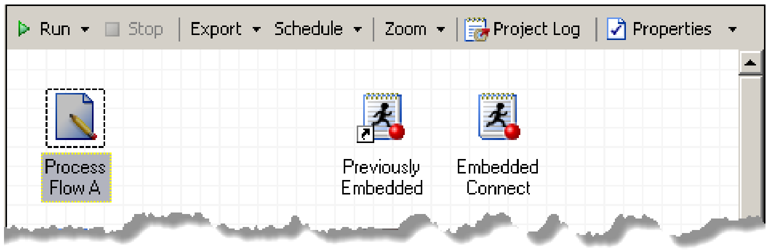
Label: External Connect
Code will run on server: Local
Last Execution Time: Unknown
File path: C:_CDMS\MWSUG 2011\EG\External Connect.sas
Location: My Computer



Label: Connect
Code will run on server: Local
Last Execution Time: Unknown
File path: (Embedded In Project)
Location: Project

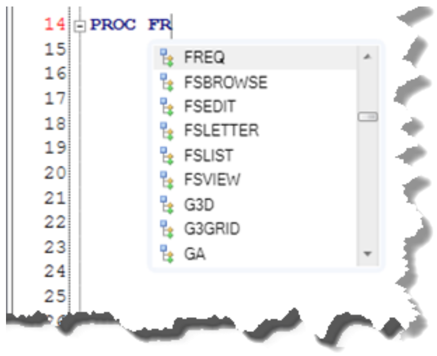
Specifies the name of the code in the Project Explorer and Project Designer. You can change the name of the code.

The result of the preceding two operations is that "Connect" is now a Linked Program ("Previously Embedded") and "External Connect" is now embedded in the Project (as "Embedded Connect").

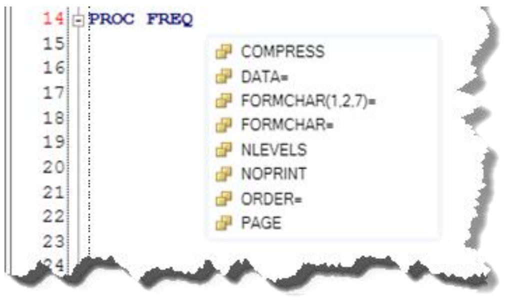


In Enterprise Guide 4.3, once you have written the code to establish libraries that point to your relational database management system, you can move the program(s) used to make these connections to a new Process Flow, name that Process Flow "Autoexec", and the code containing the LIBNAME statements will be executed automatically each time you open the project. As described by Bangi, Hemedinger, and Slocum (2010), there can be only one **Autoexec Process Flow** in any given project, but it may contain SAS Programs and/or any other EG Tasks that you want to execute as preprocessing steps prior to the execution of the remainder of the project.

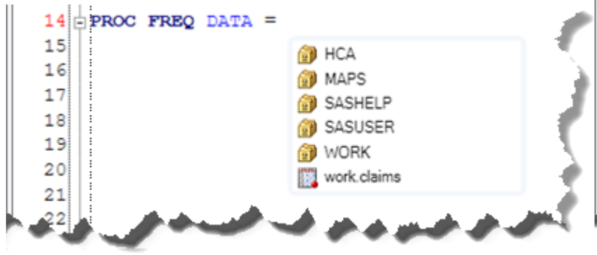
ENHANCEMENTS TO THE ENHANCED EDITOR. Regardless of whether SAS Programs are linked or embedded, the SAS Program Editor in Enterprise Guide is actually a SAS programming tool that significantly improves upon the Enhanced Editor available in the SAS software. As described in detail elsewhere [Bangi, Hemedinger, and Slocum (2010); Fecht and Dhillon (2011); and Ravenna (2011)], the version of the **Enhanced Editor** that is available in EG 4.3 provides a number to tools developed specifically for the SAS programmer. Many of the frustrations encountered by SAS programmers are overcome by the enhancements provided in EG 4.3. Among these enhancements is the ability for the Enhanced Editor to **Auto-Complete** for keywords, PROCs, and available libraries and datasets. As shown in the following example, once connected to the HCA library, you might want to run a PROC FREQ on variables in the dataset "hca.claims_2011_08". After typing the PROC keyword, the editor shows you options for auto-completing the phrase. After choosing FREQ from the list, procedure options, libraries, and datasets can be selected, in turn, using auto-complete.



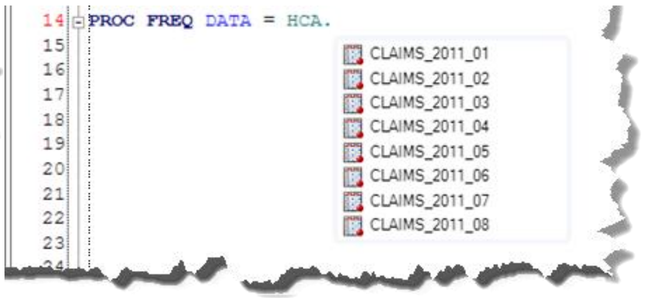
```
PROC FR
```



```
PROC FREQ
```

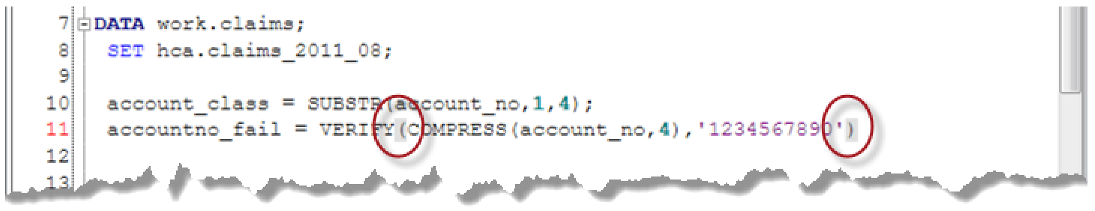


```
PROC FREQ DATA =
```



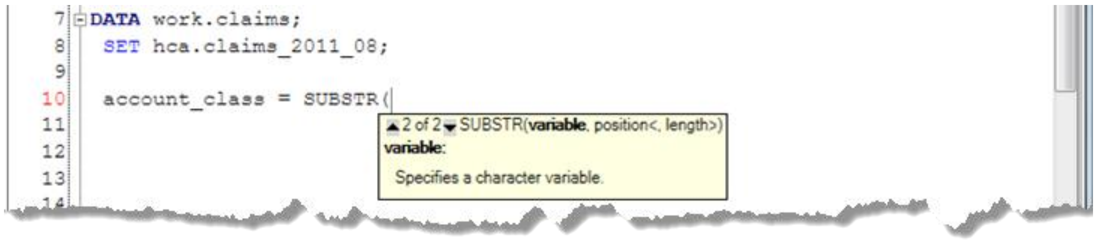
```
PROC FREQ DATA = HCA.
```

In addition to the auto-complete functionality, the EG 4.3 Enhanced Editor performs **Parentheses Matching**. If you ever write expressions with nested functions, you can appreciate how helpful this feature can be.



```
DATA work.claims;
 SET hca.claims_2011_08;

 account_class = SUBSTR(account_no,1,4);
 accountno_fail = VERIFY(COMPRESS(account_no,4),'123456789O')
```

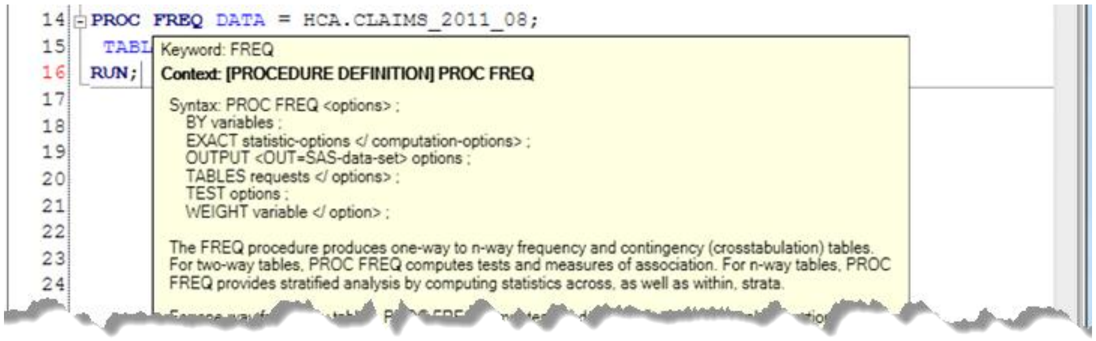
Beyond these enhancements to facilitate the mechanics of SAS programming, the EG 4.3 Enhanced Editor also provides programming support in the form of **Integrated Syntax Help** and **Function Completion**. Function completion presents users with the syntax that is available for a given function, and after the desired form of the function syntax is selected, assists the user by providing a template of the selected function syntax and provides hints as to the purpose of each argument within the syntax.



```
DATA work.claims;
 SET hca.claims_2011_08;

 account_class = SUBSTR(
```

▲2 of 2▼ SUBSTR(**variable**, position<, length>)
variable:
Specifies a character variable.

Similarly, integrated syntax help provides context-specific help topics related to the keywords being typed in a PROC or DATA step or OPTIONS statement. By simply hovering over the keyword, the user is presented with help information related to that keyword.



```
PROC FREQ DATA = HCA.CLAIMS_2011_08;
  TABL
RUN;
```

Keyword: FREQ

Context: [PROCEDURE DEFINITION] PROC FREQ

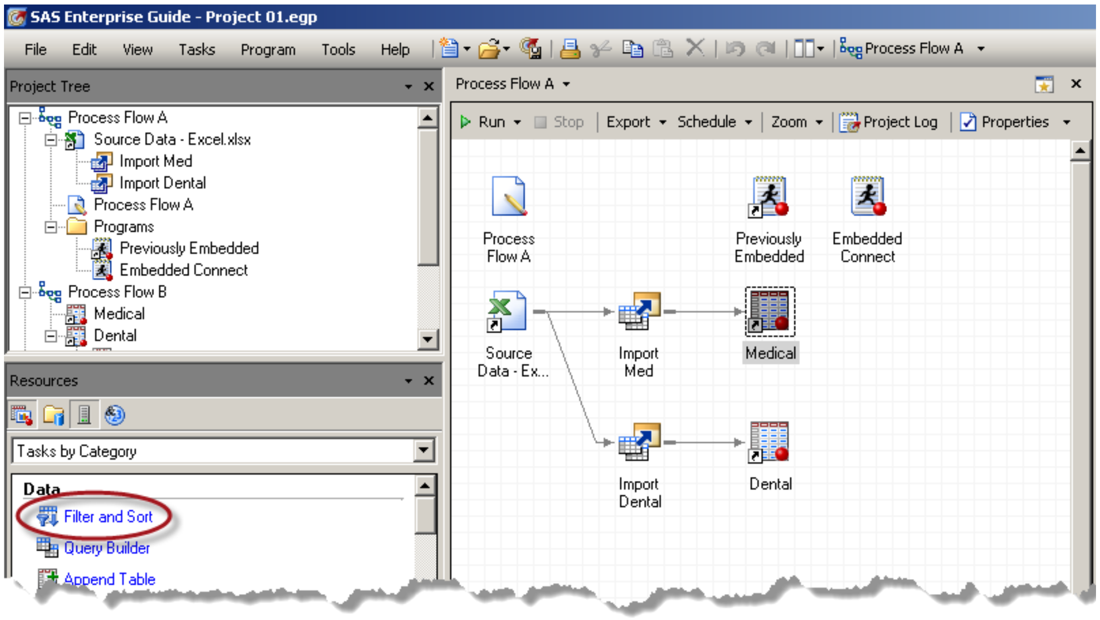
```
Syntax: PROC FREQ <options> ;
   BY variables ;
   EXACT statistic-options </ computation-options> ;
   OUTPUT <OUT=SAS-data-set> options ;
   TABLES requests </ options> ;
   TEST options ;
   WEIGHT variable </ option> ;
```

The FREQ procedure produces one-way to n-way frequency and contingency (crosstabulation) tables. For two-way tables, PROC FREQ computes tests and measures of association. For n-way tables, PROC FREQ provides stratified analysis by computing statistics across, as well as within, strata.

Of course, like most options in SAS, you can choose to turn these (and many other) options On or Off by specifying your preferences in the "Options" menu under "SAS Programs" in the "Tools" menu bar (Tools►Options►SAS Programs). Regardless of which options you find useful, however, it is clear that "the new features in the 4.3 version represent a big leap for productivity with the SAS language and programmer workflow." (Bangi, Hemedinger, and Slocum, 2010, p. 10.). Even if you choose to eschew the other features and functions of Enterprise Guide and want to simply continue to write SAS programs from scratch, EG now provides a number of options to enhance the efficiency of that work.

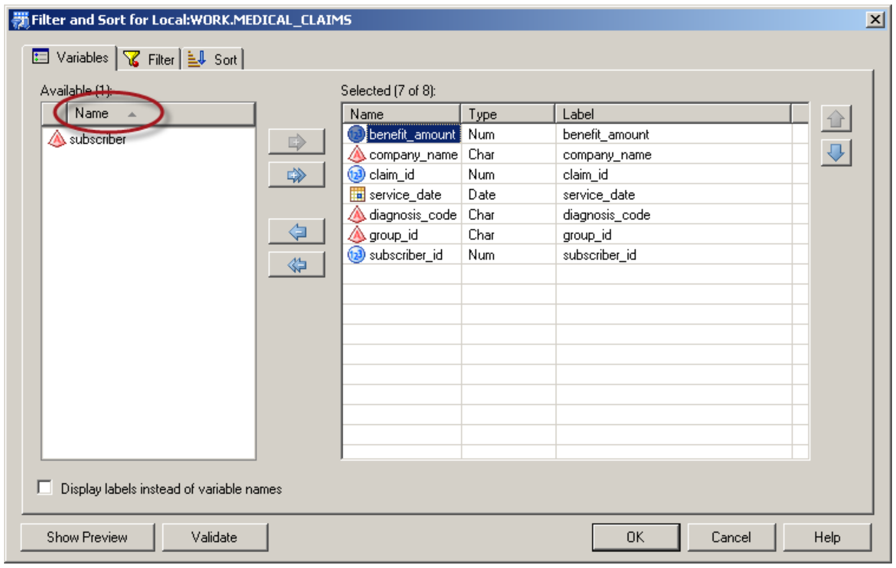
MANIPULATING DATA. Beyond these enhancements to the Enhanced Editor, however, EG offers a wide variety of Tasks that can make preparing and analyzing datasets much simpler than writing the code from scratch—even with these new enhancements to the Enhanced Editor.

Whether accessing the data for your project is done by dragging and dropping files or by referencing data in SAS libraries, one of the fundamental activities for which you will be using EG is preparing data for analysis and reporting. This can include everything from sorting data for analysis across BY groups to filtering data to produce the desired analytic subset or merging and transforming data using complex SQL statements. For each of these tasks (as well as for transposing, appending, and comparing datasets), EG provides a Task to achieve the desired outcome. In the following example, the SORT & FILTER task is used to limit a healthcare claims dataset to only those claims from Company XYZ and to sort the data by the type of coverage held by the health plan subscriber:



Like many of the tasks in EG, the user interface for specifying the Filter and Sort Task has several useful features. First, when the task is added to the process flow (by either clicking it in the Task resource or by choosing it from the "Tasks" menu bar), the currently selected (or most recently selected) dataset is used to populate the task's user interface.

The task also affords the ability to alphabetically sort the names of the available variables, which aids in selection of the variables to be included in the resulting dataset. Variables can be selected for inclusion in the resulting dataset one-by-one by selecting the variable and clicking the single arrow , by selecting multiple variables and clicking the double-arrow , or by dragging one or more variables from the "Available" list to the "Selected" list.


Filter and Sort for Local:WORK.MEDICAL_CLAIMS
Variables
Filter
Sort
Display labels instead of variable names
Show Preview
Validate
OK
Cancel
Help

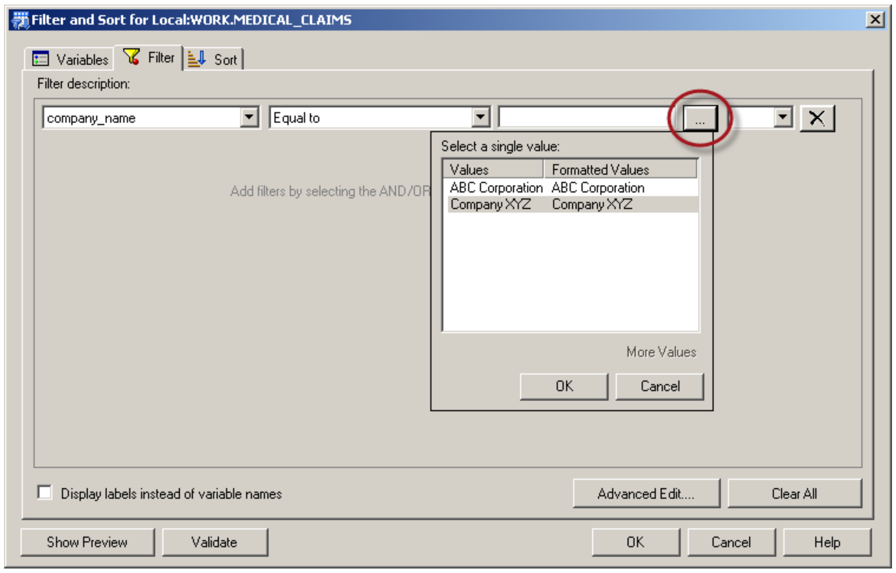
Available (1):

Name
subscriber

Selected (7 of 8):

Name	Type	Label
benefit_amount	Num	benefit_amount
company_name	Char	company_name
claim_id	Num	claim_id
service_date	Date	service_date
diagnosis_code	Char	diagnosis_code
group_id	Char	group_id
subscriber_id	Num	subscriber_id

After selecting the variables for inclusion, move to the Filter tab to establish the criteria by which you want to subset your data. The filter tab allows you to build dataset filters by specifying the variable to be evaluated, the evaluation statement, a criterion value (or values), and an operator (and/or) to build complex filter criteria. One particularly useful component of the Filter interface is the ellipsis button . When you click on the ellipsis button, you are presented with all of the distinct values for the selected variable found in the first 100,000 rows of the dataset. You can then select your criterion value from this list, and it is added to the filter expression.

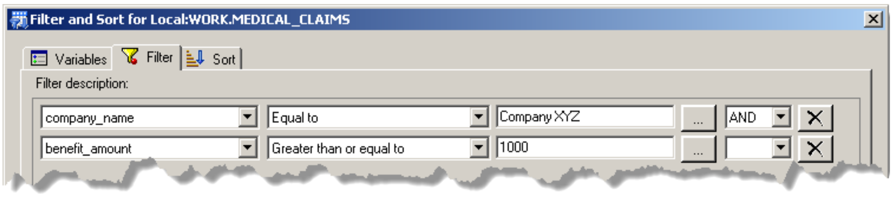

Filter and Sort for Local:WORK.MEDICAL_CLAIMS
Variables
Filter
Sort
Filter description:
Add filters by selecting the AND/OR
More Values
OK
Cancel
Display labels instead of variable names
Advanced Edit...
Clear All
Show Preview
Validate
OK
Cancel
Help

company_name | Equal to | ...

Select a single value:

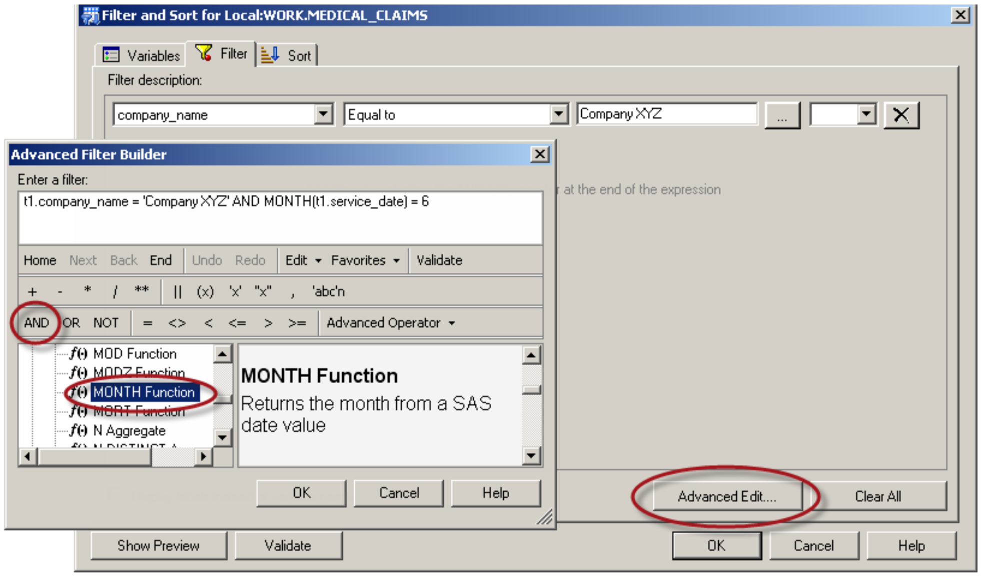
Values	Formatted Values
ABC Corporation	ABC Corporation
Company XYZ	Company XYZ

To add a second filter criterion, add an "And/Or" operator to the filter and enter the next filter criterion.



company_name	Equal to	Company XYZ	AND
benefit_amount	Greater than or equal to	1000	

To build more complex criteria involving SAS functions, algebraic expressions, or advanced operators, click on the "Advanced Edit" button to navigate to the Advanced Filter Builder. In the following example, an "AND" operator is added to the set of conditions that define the filter and the MONTH function is used to specify that the filter should also include a restriction to only select those records where the date of service provided is "June". The full complement of SAS functions is available within the Advanced Filter Builder, and when you select a function, the associated Help syntax is presented in the lower right pane of the window. Together, the Filter tab and the Advanced Filter Builder allow you to build very sophisticated filter conditions in your Filter and Sort Task.


Filter and Sort for Local:WORK.MEDICAL_CLAIMS

Filter description:

company_name | Equal to | Company XYZ

Advanced Filter Builder

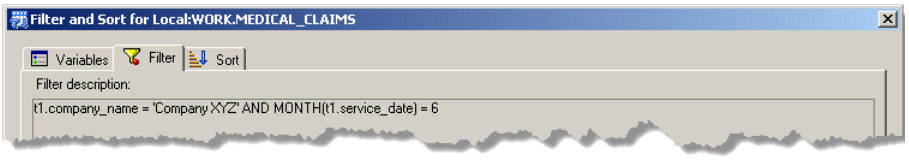
Enter a filter:

t1.company_name = 'Company XYZ' AND MONTH(t1.service_date) = 6

MONTH Function

Returns the month from a SAS date value

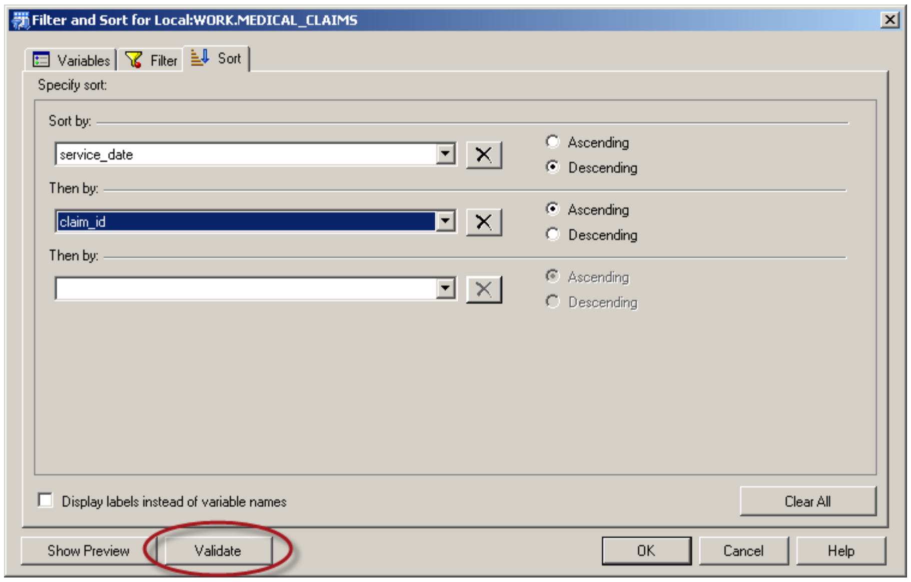
After utilizing the Advanced Editor, however, your ability to alter your point-and-click filter criteria is revoked and the filter must be edited within the Advanced Editor.


Filter and Sort for Local:WORK.MEDICAL_CLAIMS

Filter description:

t1.company_name = 'Company XYZ' AND MONTH(t1.service_date) = 6

After selecting the variables for inclusion and building your filter logic, the Sort tab can be used to order the records in the resulting dataset. In the current example, the dataset will be sorted in descending order of the value of "service_date" and (within a given value of "service_date") in ascending order of the values of "claim_id".


Filter and Sort for Local:WORK.MEDICAL_CLAIMS

Specify sort:

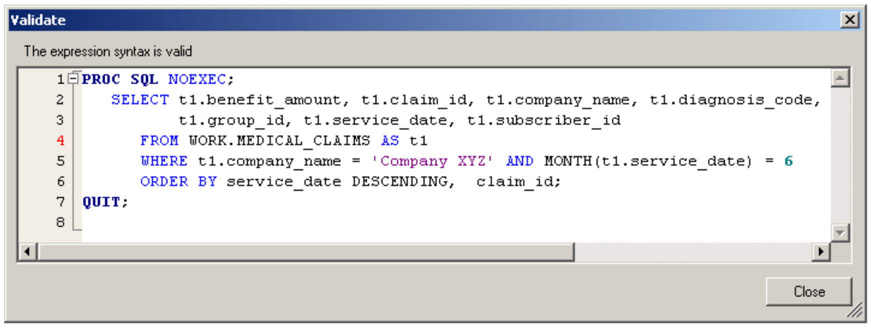
Sort by: service_date — Ascending [✓ Descending]

Then by: claim_id — [✓ Ascending] Descending

Then by: — Ascending Descending

Display labels instead of variable names

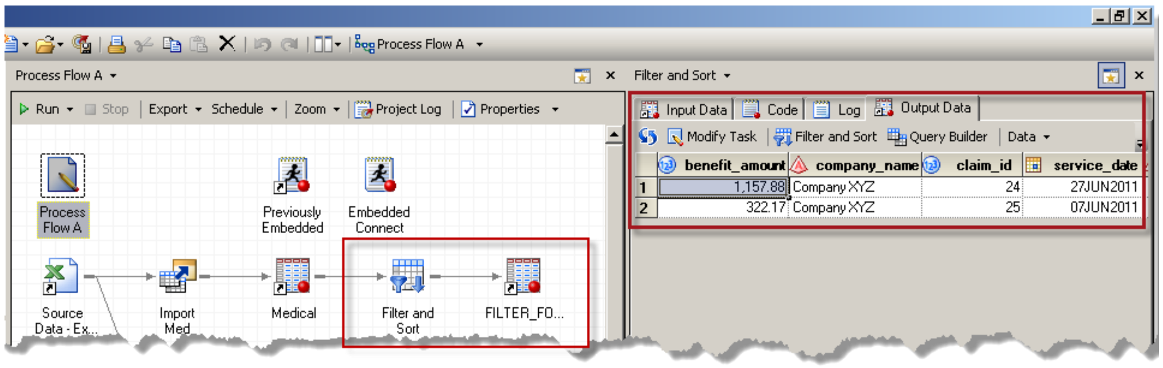
With the filter criteria and sort order specified, you can take a look at the SAS code that will be executed as a result of the task specifications by clicking on the Validate button:



The expression syntax is valid

```
PROC SQL NOEXEC;
   SELECT t1.benefit_amount, t1.claim_id, t1.company_name, t1.diagnosis_code,
          t1.group_id, t1.service_date, t1.subscriber_id
      FROM WORK.MEDICAL_CLAIMS AS t1
      WHERE t1.company_name = 'Company XYZ' AND MONTH(t1.service_date) = 6
      ORDER BY service_date DESCENDING,  claim_id;
QUIT;
```

After running the Filter and Sort Task, the resulting dataset is generated, and the new Task is added to the Process Flow.

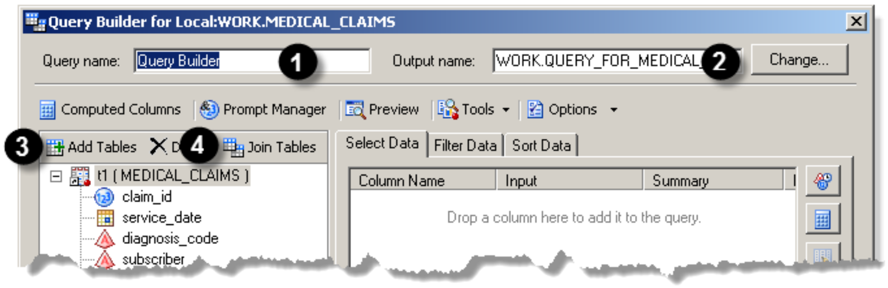


	benefit_amount	company_name	claim_id	service_date
1	1,157.88	Company XYZ	24	27JUN2011
2	322.17	Company XYZ	25	07JUN2011

As described earlier, Enterprise Guide tasks provide a graphical tool for building SAS code associated with PROC and DATA steps. The "Validate" example, above, serves to reinforce this point; the result of the point-and-click specification of the Filter and Sort Task generated the SAS code necessary to achieve the goal of filtering and sorting the source dataset. Before the experienced programmer dismisses this as a "cheat", consider, first of all, the typing (and avoidance of frustrating, time-consuming typos) that was saved by using this task. Moreover, note how easy it is to go back and rearrange the order in which the variables appear in the resulting dataset using the variable selection tab. These features, alone, make SAS EG a valuable addition to your SAS toolkit. Learning the individual tasks takes some time to be sure (just as learning new PROCs did), but the time-savings in creating and recreating datasets to suit ones needs is definitely worth the minimal time necessary to master the tasks—especially when you already know the PROCs on which they are based.

THE QUERY BUILDER TASK. One of the most important EG tasks for both developers and end-users to master for data manipulation is the QUERY task. As with the Filter and Sort task, the Query Builder task defaults to inclusion of the dataset that was most recently selected or created in the current Process Flow. Unlike some of the other tasks, however, the Query Builder task can perform a wide variety of different types of dataset transformations—including the joining of datasets, computation of new variables, and recoding of existing variables.

Upon entry into the Query Builder task, you have the ability to (1) assign the task a name to identify it in the Process Flow, (2) assign the name of the output dataset, (3) add tables to the query so that variables from those tables can be included in the output dataset and/or used to filter and order the rows in the result set, and (4) specify the join condition that will be used to associate records from each of the included tables.

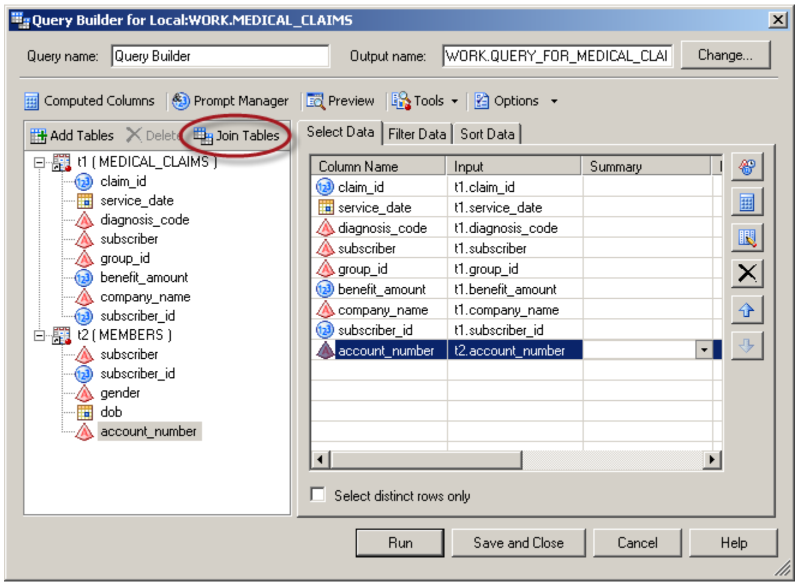


Query name: Query Builder

Output name: WORK.QUERY_FOR_MEDICAL

- t1 (MEDICAL_CLAIMS)
  - claim_id
  - service_date
  - diagnosis_code
  - subscriber

In the following example, we add the account number from a health plan's membership table to our query of the healthcare claims data. The first step is to add the members dataset to the Tables pane by clicking "Add Tables" and navigating to the members dataset. Once the members dataset is added to the query, we can select the "account_number" variable by simply dragging and dropping onto the Select Data tab.

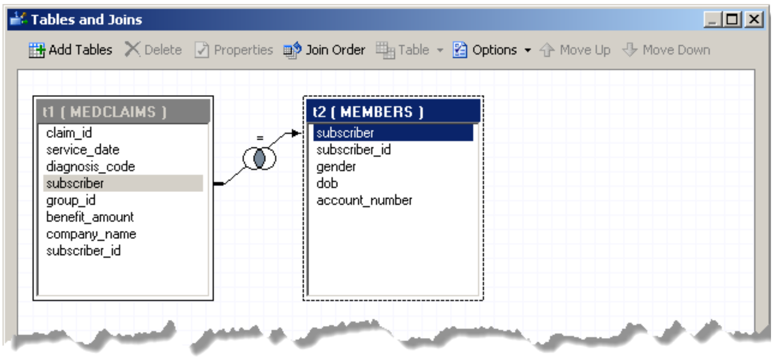


Query name: Query Builder

Output name: WORK.QUERY_FOR_MEDICAL_CLAI

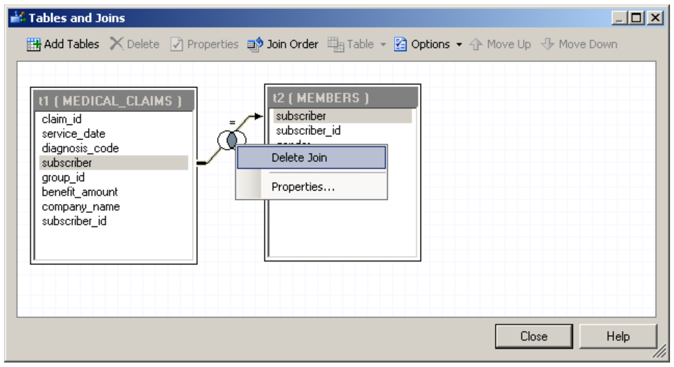
Column Name	Input	Summary
claim_id	t1.claim_id	
service_date	t1.service_date	
diagnosis_code	t1.diagnosis_code	
subscriber	t1.subscriber	
group_id	t1.group_id	
benefit_amount	t1.benefit_amount	
company_name	t1.company_name	
subscriber_id	t1.subscriber_id	
account_number	t2.account_number	

To specify how the tables should be joined, click on "Join Tables" to bring up the **Tables and Joins** window.

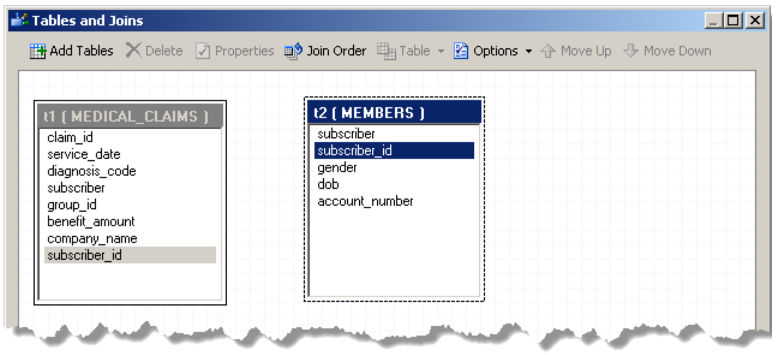


By default, Query Builder will attempt to determine which fields should be used to join the tables specified in the query. In this case Query Builder has detected that both datasets contain the variable "subscriber" and assumes that you want to perform an equi-join between the members and their healthcare claims. However, in the current example, "subscriber_id" is the field on which the tables should be joined, and the goal of the join is to determine the total amount of claims paid per health plan member, so a left-join of members to claims will be performed [see Lafler (2004, 2005) or Schacherer & Detry (2010) for an in-depth treatment of SQL joins in SAS].

First, the existing join is deleted by right-clicking on it and choosing "Delete Join".



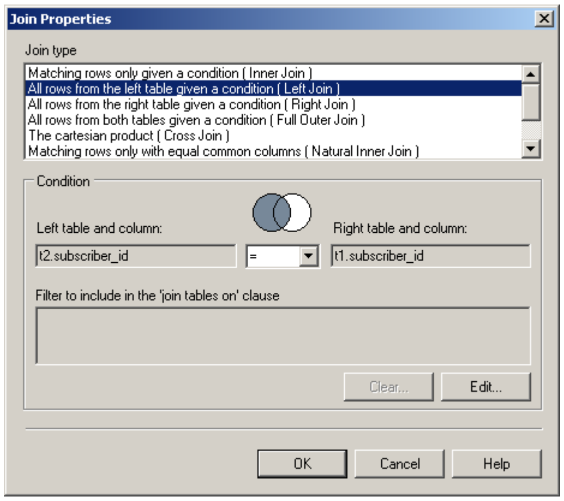
Then the new join is created by dragging "subscriber_id" from the members table onto "subscriber_id" on the medical_claims table.



t1 (MEDICAL_CLAIMS)
- claim_id
- service_date
- diagnosis_code
- subscriber
- group_id
- benefit_amount
- company_name
- subscriber_id

t2 (MEMBERS)
- subscriber
- subscriber_id
- gender
- dob
- account_number

Dragging the joined filed from one table to another, invokes the **Join Properties** dialog box. By choosing the Join Type "Left Join", we specify that we want the query to return all rows in the "left" table (i.e., the table from which subscriber_id was first selected—members) and only those rows from the medical_claims table that contain a record with a matching subscriber_id.



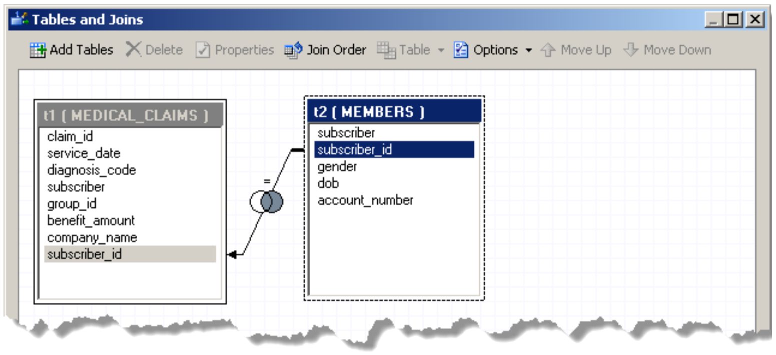
Join type
- Matching rows only given a condition (Inner Join)
- All rows from the left table given a condition (Left Join)
- All rows from the right table given a condition (Right Join)
- All rows from both tables given a condition (Full Outer Join)
- The cartesian product (Cross Join)
- Matching rows only with equal common columns (Natural Inner Join)

Condition

Left table and column: t2.subscriber_id = Right table and column: t1.subscriber_id

Filter to include in the 'join tables on' clause

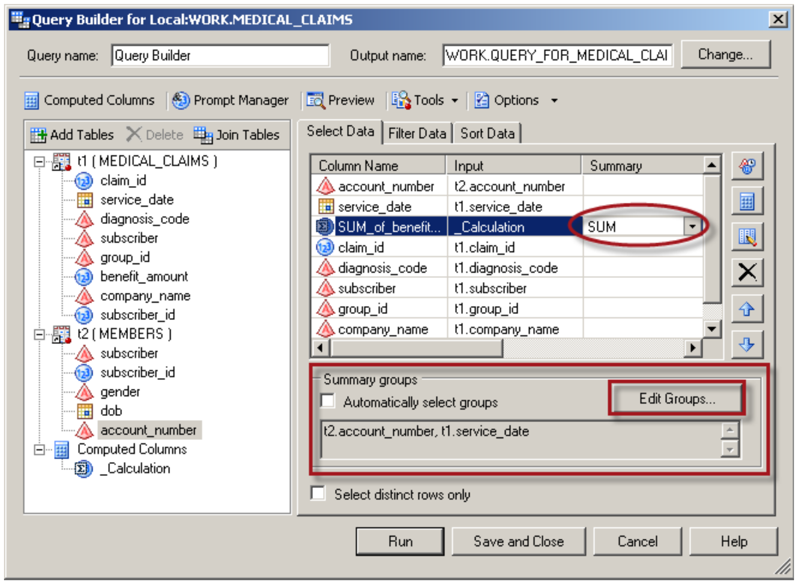
Once the new join type is selected, click OK to assign the new join condition and close the Tables and Joins window to return to the Query Builder's main interface.



t1 (MEDICAL_CLAIMS)
- claim_id
- service_date
- diagnosis_code
- subscriber
- group_id
- benefit_amount
- company_name
- subscriber_id

t2 (MEMBERS)
- subscriber
- subscriber_id
- gender
- dob
- account_number

Upon returning to the Query Builder's main window, you can rearrange the order of the variables in the Select Data tab using the up and down arrows to move the column names up and down in the presentation order (note the order of "t2.account_number" and "t1.service_date"). Additionally, you can add summary values by choosing a summary function from the "Summary" column's drop-down list. In this example, we are going to sum the value of "benefit_amount" across each unique combination of "account_number" and "service_date". By default, when a summary function is specified "Automatically select groups" is checked for you, with all other selected variables defining the group-by term. To redefine the group-by clause, uncheck the box and click "Edit Groups" to be taken to a pop-up that allows you to select the variables used to define your grouping term.



Query name: Query Builder
Output name: WORK.QUERY_FOR_MEDICAL_CLAI

Column Name	Input	Summary
account_number	t2.account_number	
service_date	t1.service_date	
SUM_of_benefit...	_Calculation	SUM
claim_id	t1.claim_id	
diagnosis_code	t1.diagnosis_code	
subscriber	t1.subscriber	
group_id	t1.group_id	
company_name	t1.company_name	

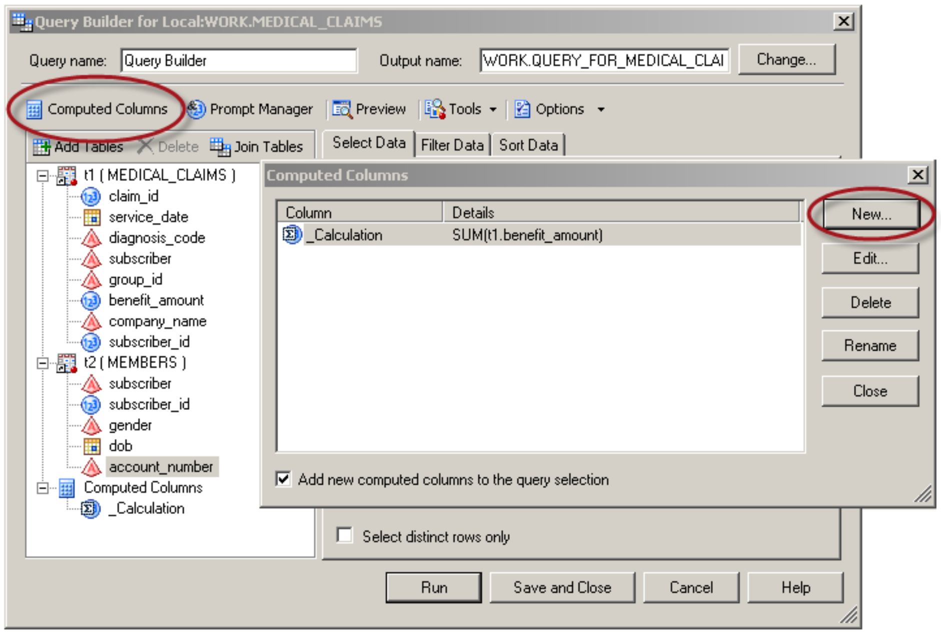
Summary groups
☐ Automatically select groups
t2.account_number, t1.service_date

☐ Select distinct rows only

To change the Sort Order (ORDER BY clause) or to add a Filter Condition (to the WHERE clause), the "Filter" and "Sort" tabs provide the same highly intuitive interfaces demonstrated in the earlier example of the Filter and Sort Task.

In addition to joining and summarizing data, however, you often need to create or transform variables in ways other than those available through the SQL language's aggregation functions. In PROC SQL, you might accomplish these tasks by using a CASE statement to perform recoding based on conditional logic, by writing an arithmetic expression, or through the application of SAS Functions. In the Query Task all such transformations are achieved using **Computed Columns**.

In order to create a Computed Column, click on "Computed Columns" button on the Query Builder Task's main screen and click "New" in the Computed Columns window.



Query name: Query Builder
Output name: WORK.QUERY_FOR_MEDICAL_CLAI

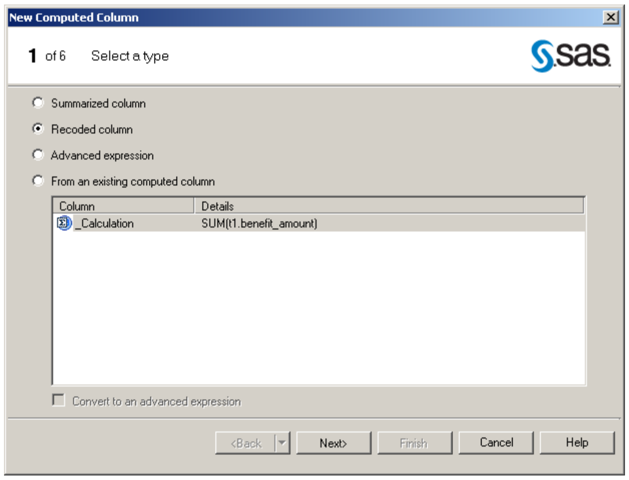
Computed Columns

Column	Details
_Calculation	SUM(t1.benefit_amount)

☑ Add new computed columns to the query selection

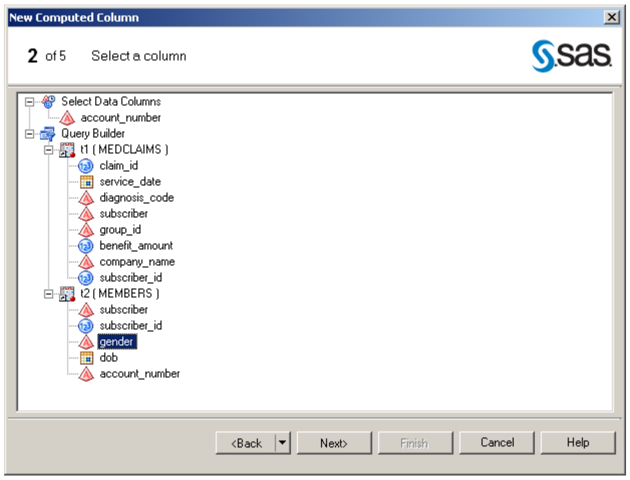
☐ Select distinct rows only

Next, select the Computed Column Type. Summarized columns are those that are computed using SQL aggregate functions. Advanced expressions are those that use the Advanced Expression editor similar to the Advanced Filter Builder in the previous Filter and Sort example, and Recoded columns are those that assign values based on some logical condition evaluated for each record. Finally, columns produced "From an existing computed column" are those for which an existing computed column is used as the basis for the computation used to produce the new column. In the following example, the "Recoded column" type is used to convert values of "M" and "F" in the "gender" column of the "members" dataset to the values "Male" and "Female", respectively, in the variable "member_gender".

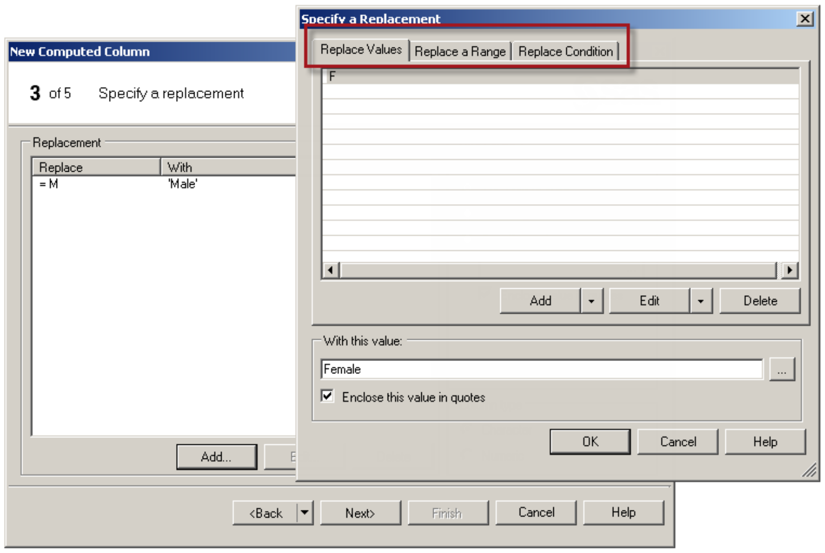


Column	Details
_Calculation	SUM(t1.benefit_amount)

After selecting the computed column type and clicking "Next", the gender column is selected as the basis for the recoded variable. Once the column is selected, click "Next" again to advance to the specification of the values to be recoded.



The values to be replaced are then specified along with their recoded values. Note that a number of replacement strategies are supported. One can specify individual values to be replaced (e.g., "M" recoded as "Male"), ranges of values can be replaced (e.g., ages 0 – 17 recoded as "Child"), or recoding can be based on a number of other conditional logic operators (e.g., "claim_type" NOT IN 1,3,4,7,10 recoded as "Other").



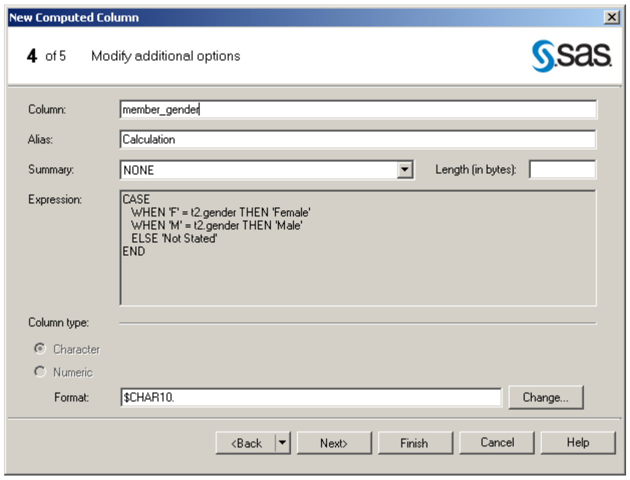
Replace	With
= M	'Male'

F

With this value: Female

☑ Enclose this value in quotes

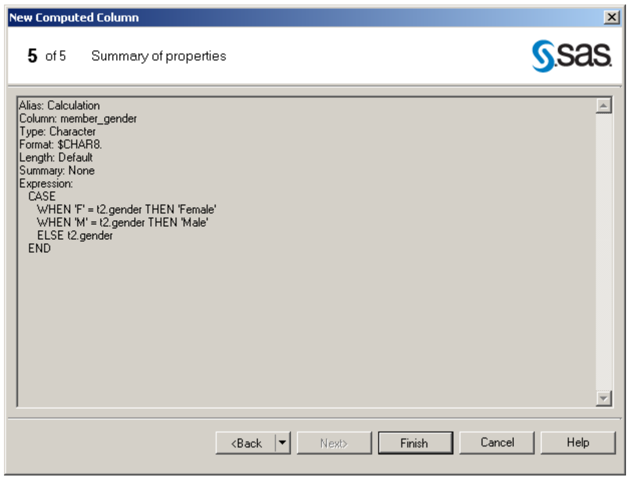
In the last functional step of recoding a variable, you supply a variable name for the new computed variable and assign a format for the column. At this step, you can also see the syntax of the CASE statement that will be generated by the Query Builder task.


New Computed Column

4 of 5 Modify additional options

Column: member_gender
Alias: Calculation
Summary: NONE
Length (in bytes):
Expression:
```
CASE
  WHEN 'F' = t2.gender THEN 'Female'
  WHEN 'M' = t2.gender THEN 'Male'
  ELSE 'Not Stated'
END
```
Column type:
[✓ Character] Numeric
Format: $CHAR10.

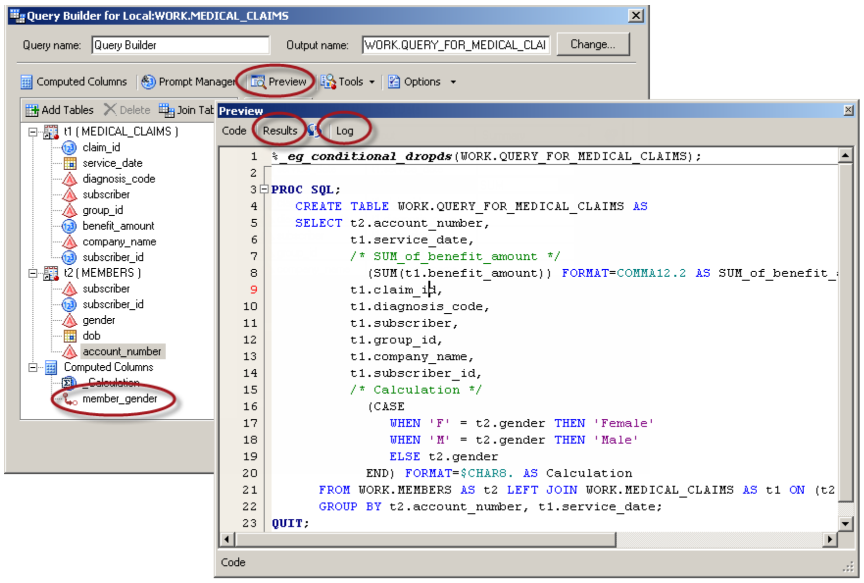
Finally, in Step 5, you are presented with a summary of the properties for the new computed variable and you can either click "Finish" to complete the creation of the query syntax and return to the Query Builder screen or click "Back" to go step back through the **New Computed Column** wizard and alter the specifications for the new column.


New Computed Column

5 of 5 Summary of properties

```
Alias: Calculation
Column: member_gender
Type: Character
Format: $CHAR8.
Length: Default
Summary: None
Expression:
  CASE
    WHEN 'F' = t2.gender THEN 'Female'
    WHEN 'M' = t2.gender THEN 'Male'
    ELSE t2.gender
  END
```

As in the previous Filter and Sort example, you can preview the SAS code that will be generated by the Query Task before running it. Click "Preview" on the Query Task window and a Preview window containing the associated SAS code is presented. You can also, preview the results of the query and check the log for any syntax errors that will arise from running the task.


Query Builder for Local:WORK.MEDICAL_CLAIMS

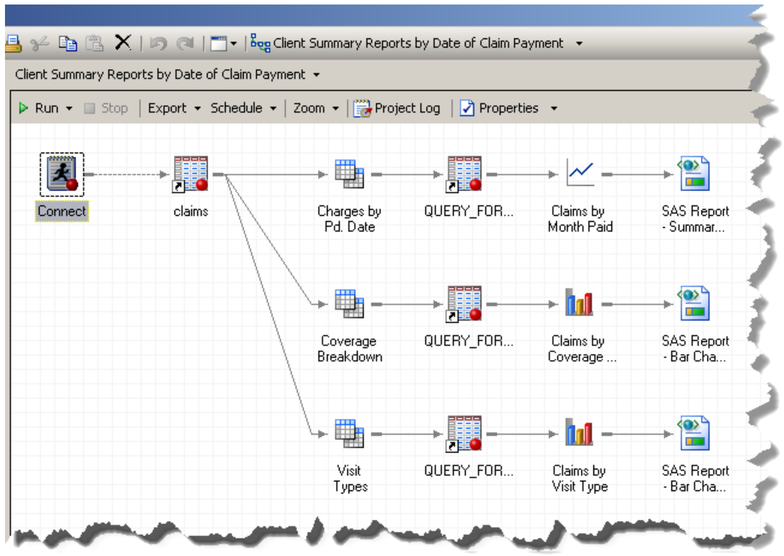
Query name: Query Builder
Output name: WORK.QUERY_FOR_MEDICAL_CLAI

```
%_eg_conditional_dropds(WORK.QUERY_FOR_MEDICAL_CLAIMS);

PROC SQL;
   CREATE TABLE WORK.QUERY_FOR_MEDICAL_CLAIMS AS
   SELECT t2.account_number,
          t1.service_date,
          /* SUM_of_benefit_amount */
            (SUM(t1.benefit_amount)) FORMAT=COMMA12.2 AS SUM_of_benefit_
          t1.claim_id,
          t1.diagnosis_code,
          t1.subscriber,
          t1.group_id,
          t1.company_name,
          t1.subscriber_id,
          /* Calculation */
            (CASE
               WHEN 'F' = t2.gender THEN 'Female'
               WHEN 'M' = t2.gender THEN 'Male'
               ELSE t2.gender
            END) FORMAT=$CHAR8. AS Calculation
      FROM WORK.MEMBERS AS t2 LEFT JOIN WORK.MEDICAL_CLAIMS AS t1 ON (t2
      GROUP BY t2.account_number, t1.service_date;
QUIT;
```

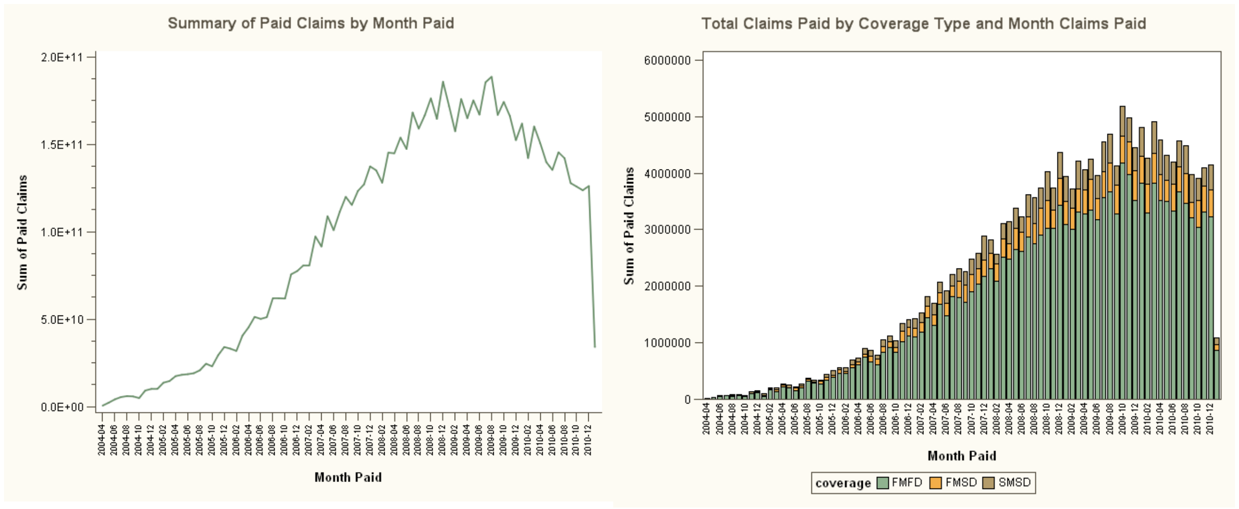
As the Query Task and Filter and Sort Task examples demonstrate, one of the main goals of EG is to enable end-users without SAS programming expertise to utilize the power of the SAS programming environment to manipulate and analyze data. One should not conclude, however, that EG obviates the need for SAS programmers; what is advocated here is that SAS programmers embrace EG as a tool that can be used to help deliver the analytic power of SAS to non-programmer end-users so that they can more efficiently use their content knowledge to help your organization remain competitive with respect to data-driven decision-making. Instead of these users coming to you each time they need to add a column to an output dataset, summarize detail data, or refresh a dataset that you produced for them as a "one-time" ad hoc, you can use SAS EG to put this capability in their hands and focus your efforts on more gratifying programming challenges such as leveraging the built-in capabilities of SAS EG to developing enterprise solutions.

PROMPTS, AUTOMATED DELIVERY, CONDITIONAL PROCESSING, AND WINDOWS SCHEDULING

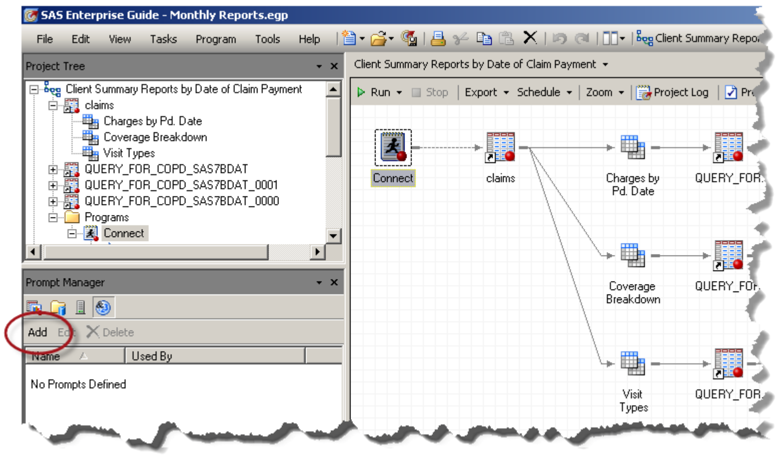
As an example of building an application that uses EG built-in capabilities to further empower your end-users, consider the following SAS EG Project "Monthly Reports". An account manager at a third party administrator of healthcare claims wanted some simple reports that would show her a few different breakdowns of claim payments for her client (Company XYZ). You quickly produce these reports with a few Query Tasks followed by Line Graph and Bar Chart Tasks.

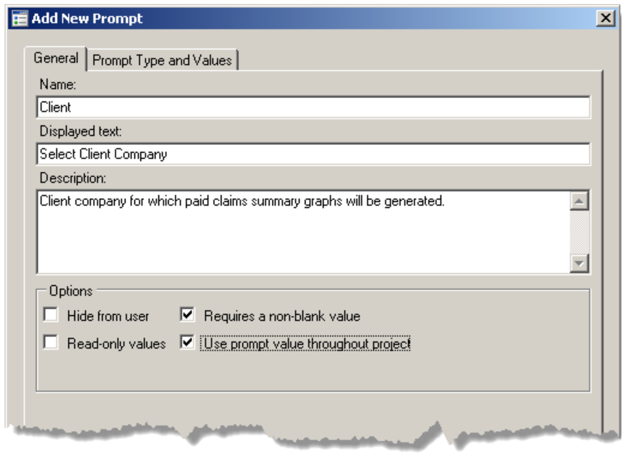


The resulting graphs are exactly what she wanted, and after she shares them with a colleague, he decides that he would like similar graphs delivered monthly for his clients. You suspect that requests for these graphs could grow rapidly as they get shared with other account managers. You want to be able to rapidly fulfill the needs of these data consumers, so you rethink the original project and realize that you need to build the queries with the flexibility to change the client company on the fly—running the same code for Company XYZ, Company ABC, or JKL Corporation.



PROMPTS. As a SAS programmer you realize immediately that there needs to be a macro variable in the WHERE clause of each query so that different company identifiers can be assigned without changing the individual Query Builder Tasks. In EG, the key to creating this flexibility is provided by **Prompts**. As mentioned earlier, Prompts facilitate user interaction with Projects by allowing users to select values from a list, enter individual values, provide lists of values, etc. Prompts are used in the following example to allow users to specify the client company for which the Weekly Reports Project will be run.





Name: Client

Displayed text: Select Client Company

Description: Client company for which paid claims summary graphs will be generated.

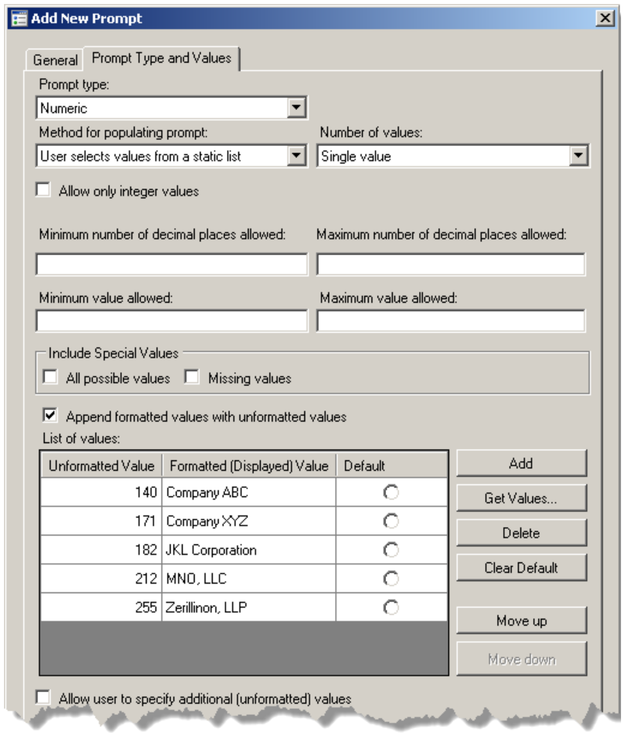
Options

☐ Hide from user ☑ Requires a non-blank value

☐ Read-only values ☑ Use prompt value throughout project

The first step in rewriting the Monthly Reports project is to add the Prompt that will be displayed to users when the project is run. The Prompt "Client" is created by clicking "Add" in the Prompt Manager resource and giving the new prompt a **Name**, **Display Text**, and a **Description**. The "client" prompt will also be required to have a non-null value and will retain its value throughout the execution of the "Monthly Reports" project.

Because the company id that is used in the WHERE clause of the project queries is numeric, the **Prompt Type** for "Client" is specified as Numeric. If you want users of this project to select their client company by choosing it from a drop-down list, choose "User selects values from a static [or dynamic] list" under **Method for populating prompt**. The **Number of values** that users of this prompt can select is a single value—their individual clients. The values for this static list can be manually entered using the **Add** button next to the List of Values, or automatically populated from a datasource using the **Get Values** button. Users of the "Client" prompt will be presented with the **Display Value** appended with the **Unformatted Value** (e.g., "Company ABC [140])—which serves as a redundant piece of information for users who might be equally familiar with the company's client code within the claims processing system as they are with the Formatted (Displayed) Value. Once the prompt is built, click the OK button to save the prompt to the project.



Prompt type: Numeric

Method for populating prompt: User selects values from a static list

Number of values: Single value

☐ Allow only integer values

Minimum number of decimal places allowed:

Maximum number of decimal places allowed:

Minimum value allowed:

Maximum value allowed:

Include Special Values

☐ All possible values ☐ Missing values

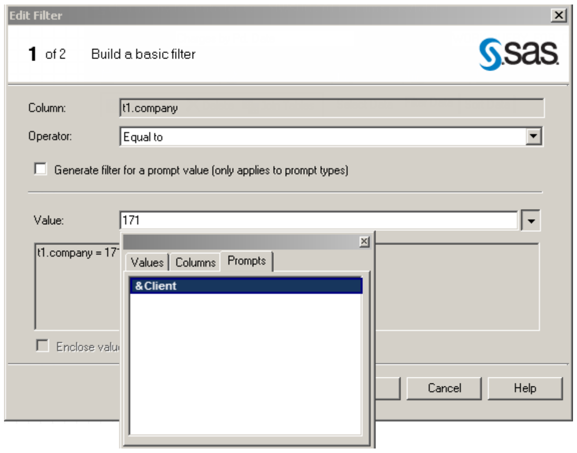
☑ Append formatted values with unformatted values

List of values:

Unformatted Value	Formatted (Displayed) Value	Default
140	Company ABC	○
171	Company XYZ	○
182	JKL Corporation	○
212	MNO, LLC	○
255	Zerillinon, LLP	○

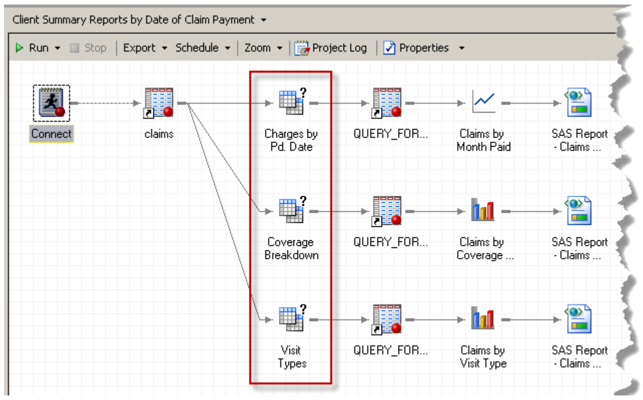
☐ Allow user to specify additional (unformatted) values

Next, the prompt needs to be associated with the Filter (or WHERE clause) of the Query Builder Task. This association is made by selecting the Query Builder Task in the Process Flow, right clicking it, and choosing **Modify <name of task>**. In the adjacent figure, the filter of the "Charges by Pd. Date" Query Builder Task is being edited. Instead of providing a specific company id (e.g., "170) as the filter condition for this query (as was done originally), navigate to the "Prompts" tab of the Value dropdown and select "&Client".

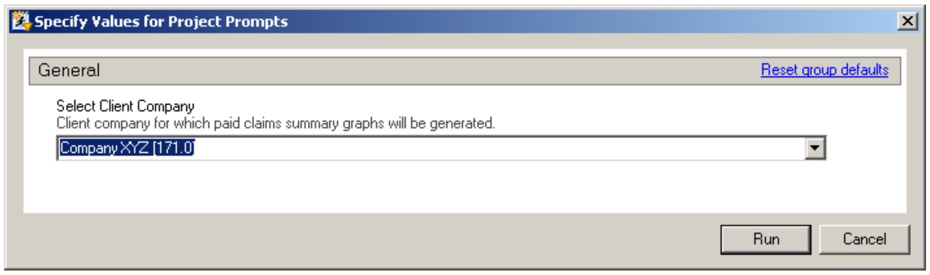


Column: t1.company
Operator: Equal to
Generate filter for a prompt value (only applies to prompt types)
Value: 171
t1.company = 17
&Client

Once all three queries are changed to filter "claims" records based on the Client Prompt instead of a specific, hard-coded value, the Process Flow is ready to be run in a manner that is driven by the response to "Client" provided by the user. [Notice how the icons representing these queries have now changed in the Process Flow.]



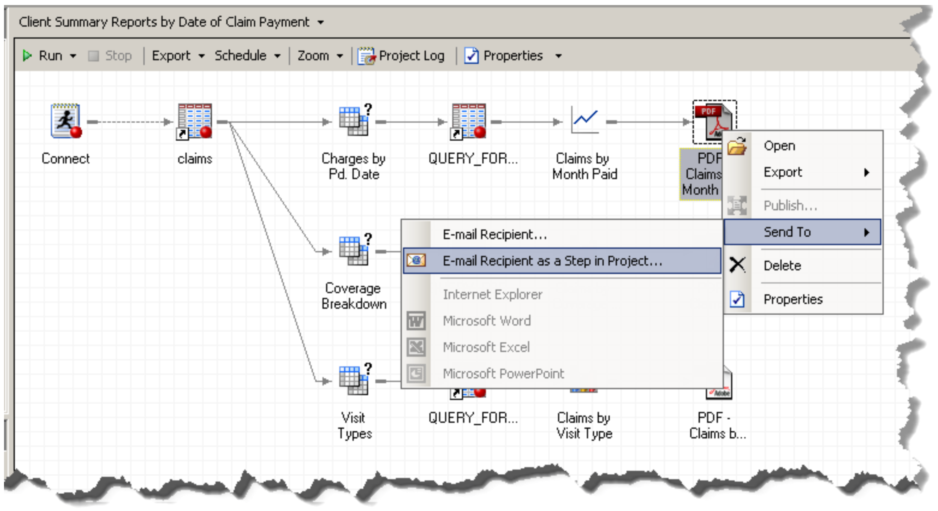
When the Process Flow is next run, the user will be presented with the Prompt that was specified earlier:



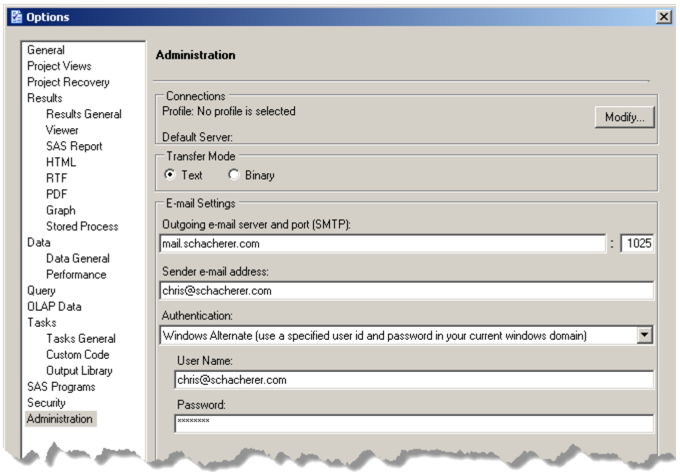
Select Client Company
Client company for which paid claims summary graphs will be generated.
Company XYZ [171.0]

Once a Prompt Value is chosen by the user, the Process Flow continues on through the Query Builder Tasks and on to the production of the Reports for that particular client company. To produce the graphs for a different company, all one needs to do is choose a different response for the prompt when the Process Flow is next run.

DELIVERY AUTOMATION. In addition to facilitating the flexible production of reports, however, SAS Enterprise Guide also has built-in functionality to facilitate the distribution of analytic output. To expand on the previous example, once the reports are generated, you could use the **Send To** functionality to send the resulting reports to a list of recipients via e-mail.



In order for this e-mail option to work, you will have to provide the configuration parameters associated with the e-mail account from which you will be sending the message. To specify this information, select Tools► Options► Administration on the menu bar and provide the required information for your mail server. Once your e-mail configuration is specified, however, selecting Send To►E-mail Recipient as a Step in Project will walk you through a three-step wizard to attach any files you are sending, specify the recipient(s) of the e-mail, and write the associated e-mail message. At that point, sending the e-mail simply becomes a task to be executed in the Process Flow.



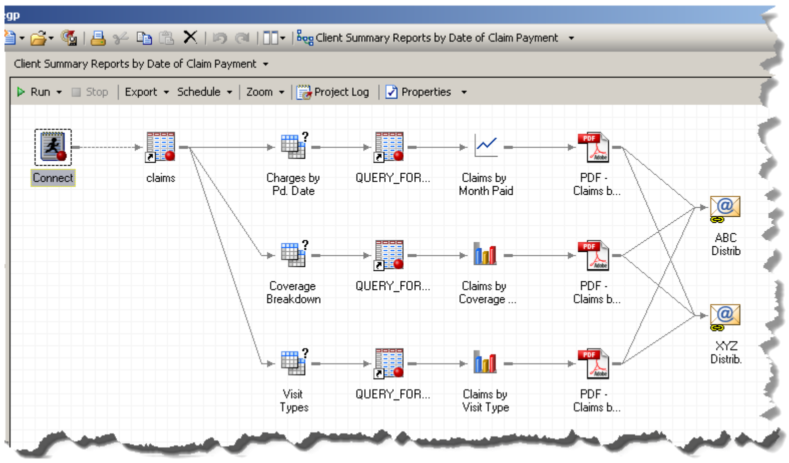
Administration

Connections
Profile: No profile is selected
Default Server:

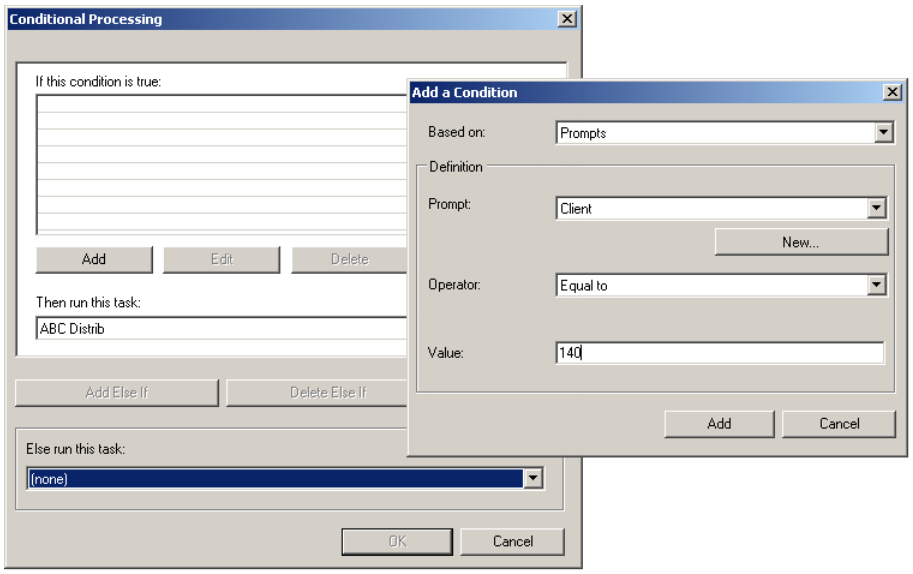
Transfer Mode
[✓ Text] Binary

E-mail Settings
Outgoing e-mail server and port (SMTP): mail.schacherer.com : 1025
Sender e-mail address: chris@schacherer.com
Authentication: Windows Alternate (use a specified user id and password in your current windows domain)
User Name: chris@schacherer.com
Password: ********

CONDITIONAL PROCESSING. One remaining challenge for the distribution of these reports is that for each client company there will likely be different distribution lists. That is, Mary might want to receive all Company XYZ reports, but not Company ABC reports, and the converse might be true for Bill. If that is the case, you can take advantage of EG's **Conditional Processing** functionality to control the distribution of reports based on the value of the "Client" Prompt chosen for each execution of the project. The first step in creating this conditional logic is to create two different mail messages—one for Company ABC (specifying Mary's e-mail address) and one for Company XYZ (specifying Bill's e-mail address). Next, a Conditional Processing step is added to the "ABC Distrib" node by right-clicking on the node and selecting Condition►Add.



The properties of this condition specify that the Condition is based on the value of a prompt. Specifically, if the value of the Prompt "Client" equals "140" the "ABC Distrib" task will be run and the reports generated in the Process Flow will be e-mailed to Bill.



Then run this task: ABC Distrib

Else run this task: (none)

Based on: Prompts

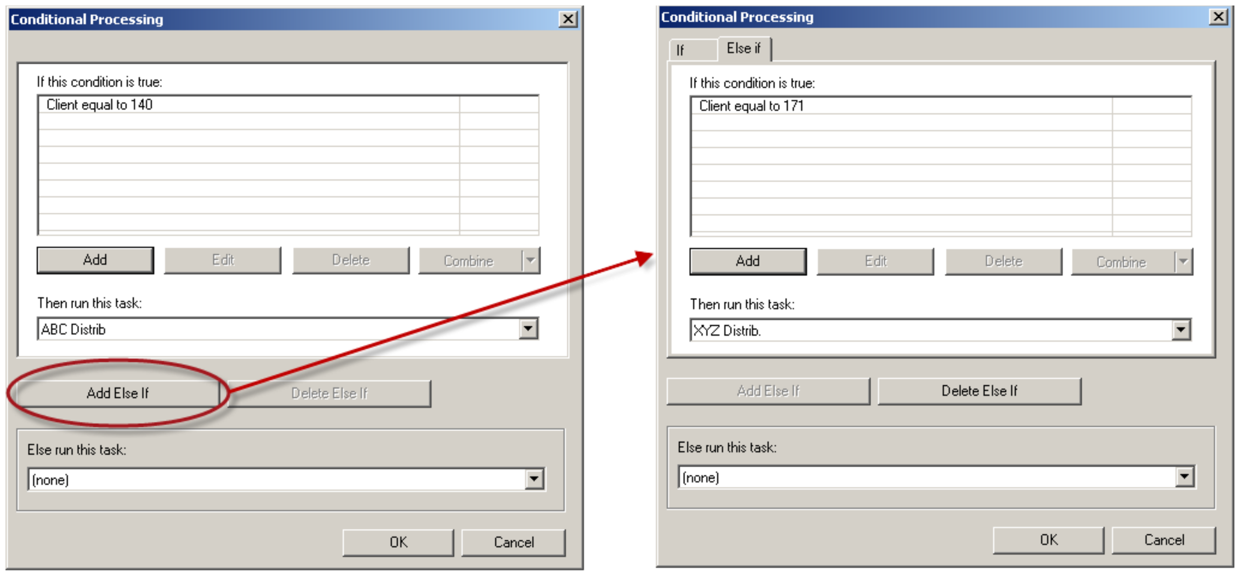
Definition

Prompt: Client

Operator: Equal to

Value: 140

Once that condition is created, you can add an "Else If" condition to assess whether "XYZ Distrib" should be executed in those cases where the first condition "&Client = 140" does not evaluate to "TRUE". If the value of "Client" is 171 instead of 140, "XYZ Distrib" will be run—sending the results to Mary.



If this condition is true: Client equal to 140

Then run this task: ABC Distrib

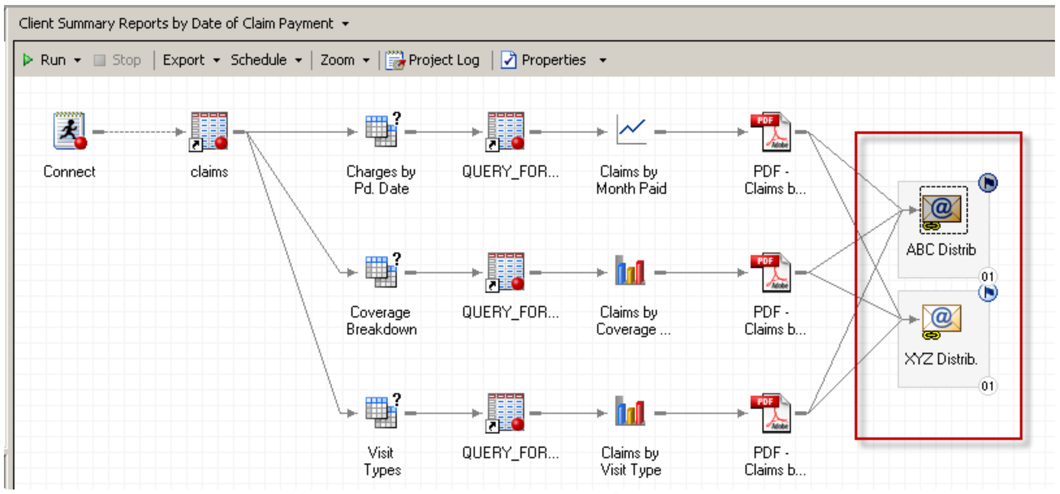
Else run this task: (none)

If this condition is true: Client equal to 171

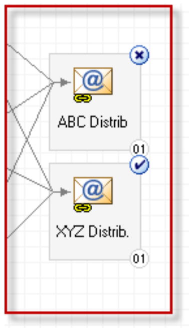
Then run this task: XYZ Distrib.

Else run this task: (none)

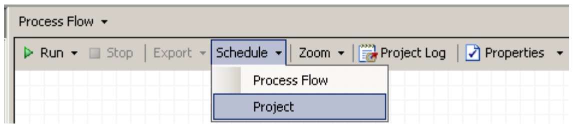
Once saved, this conditional logic is denoted in the Process Flow by the changes in the tasks "ABC Distrib" and "XYZ Distrib" that indicate they are contingent on the outcome of the new condition.

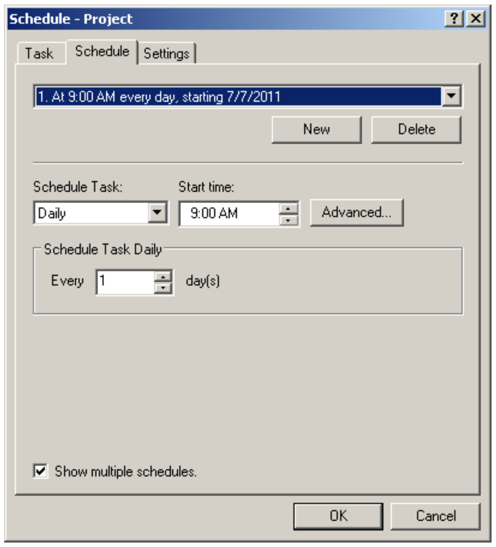


After choosing Company XYZ as the response to the Client Prompt, the representation of the distribution tasks changes once again to indicate that "ABC Distrib" did not meet the condition required for its execution, but "XYZ Distrib" did—and was run as a result of the conditional processing. Conditional Processing and Prompts are very powerful ways to control processing in SAS EG; for more in-depth information on Prompts and Conditional Processing, the reader is referred to Hall (2011) and Sucher (2010).



WINDOWS SCHEDULING. In addition to conditional processing and integrated e-mail as a means for automating data delivery, Enterprise Guide also facilitates the scheduling of Project and Process Flow execution by providing a hook into the Windows Scheduler. From within the EG interface, you can schedule your project (or an individual Process Flow within a project) to run at a specified time of day, day of week, etc. Once the schedule is specified, EG generates a VBScript that is launched by the Windows Scheduler according to the designated schedule, and this Script, in turn, launches the scheduled Project or Process Flow via Enterprise Guide.





1. At 9:00 AM every day, starting 7/7/2011

Schedule Task: Daily
Start time: 9:00 AM

Schedule Task Daily
Every 1 day(s)

☑ Show multiple schedules.

CONCLUSION

The inclusion of developer-centered functionality like Prompts, Conditional Logic, and Automated Scheduling and Deliver, as well as the integration of smart, context-sensitive auto-completion in the Enhanced Editor really serves to emphasize the fact that SAS Enterprise Guide is becoming a tool that should not be dismissed as simply a point-and-click tool for those who do not use SAS. By providing non-programmers access to the power of SAS analytic procedures, Enterprise Guide provides organizations the ability to put real analytic power in the hands of the users who are managing their business operations. This has the added benefit of freeing IT, Analytics, and Decision Support analysts from having to spend their time rerunning and tweaking reports and analyses every time a minor change is needed in formatting, column/row order, or specification of a subset of data. Freed from these duties, these programmers and analysts can put their SAS skills to work developing even more powerful analytic tools for the non-programmer users throughout your organization. This virtuous cycle of enablement can facilitate your organization's ability to turn data into information and provide an ever-broadening group of your co-workers with the information needed to help your organization gain a competitive advantage.

REFERENCES

Bangi, A., Hemedinger, C., Slocum, S. (2010). New Goodies in SAS® Enterprise Guide® 4.3. Proceedings of SAS Global Forum 2010. Cary, NC: SAS Institute, Inc.

Fecht, M. (2009). THINK Before You Type… Best Practices Learned the Hard Way. Proceedings of SAS Global Forum 2009. Cary, NC: SAS Institute, Inc.

Fecht, M. & Dhillon, R. (2011). SAS Enterprise Guide 4.3: Finally a Programmer's Tool. Proceedings of SAS Global Forum 2011. Cary, NC: SAS Institute, Inc.

Hall, A. (2011). Creating Reusable Programs by Using SAS® Enterprise Guide® Prompt Manager. Proceedings of SAS Global Forum 2011.

Hemedinger, C. (2007a). Efficient Data Access using SAS Enterprise Guide. SAS Sample 26178. Retrieved July 6, 2011 from : http://support.sas.com/kb/26/178.html

Hemedinger, C. (2007b). Optimize Data Access within SAS Enterprise Guide. Retrieved July 6, 2011 from : http://www.youtube.com/watch?v=OSTa1EUpKT8

Lafler, K.P. (2005). Manipulating Data with PROC SQL. Proceedings of the 30th Annual SAS Users Group International Meeting. Cary, NC: SAS Institute, Inc.

Lafler, K.P. (2004). PROC SQL: Beyond the Basics Using SAS. Cary, NC: SAS Institute, Inc.

Levine, F. (2001). Using SAS/ACCESS Libname Technology to Get Improvements in Performance and Optimizations in SAS/SQL Queries. Proceedings of the 26th Annual SAS Users Group International Meeting. Cary, NC: SAS Institute, Inc.

Ravenna, A. (2011). Becoming a Better Programmer with SAS® Enterprise Guide®. Proceedings of SAS Global Forum 2011. Cary, NC: SAS Institute, Inc.

Schacherer, C.W. & Westra, B.D. (2010). Introduction to SAS® for the Healthcare Analyst. Proceedings of the Midwest SAS Users Group. Cary, NC: SAS Institute, Inc.

Schacherer, C.W. & Detry, M.A. (2010). PROC SQL: from SELECT to Pass-Through SQL. Proceedings of the South Central SAS Users Group. Cary, NC: SAS Institute, Inc.

SAS Institute, Inc. (2004a). SAS/ACCESS 9.1 Supplement for Microsoft SQL Server. Cary, NC: SAS Institute, Inc.

SAS Institute, Inc. (2004b). SAS/ACCESS 9.1 Supplement for Oracle. Cary, NC: SAS Institute, Inc.

SAS Institute, Inc. (2009). SAS OnlineDoc® 9.2. Cary, NC: SAS Institute Inc.

SAS Institute, Inc. (2011). SAS® 9.2 Intelligence Platform: System Administration Guide, Second Edition. Cary, NC: SAS Institute Inc.

Slaughter, S.J. & Delwiche, L.D. (2010). The Little SAS Book for Enterprise Guide 4.2. Cary, NC: SAS Institute, Inc.

Sucher, K. (2010). Interactive and Efficient Macro Programming with Prompts in SAS® Enterprise Guide® 4.2. Proceedings of SAS Global Forum 2010. Cary, NC: SAS Institute, Inc.

CONTACT INFORMATION

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

Christopher W. Schacherer, Ph.D.
Clinical Data Management Systems, LLC
Madison, WI 53719
E-mail: CSchacherer@cdms-llc.com
Web: www.cdms-llc.com