

Paper 294-2012

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

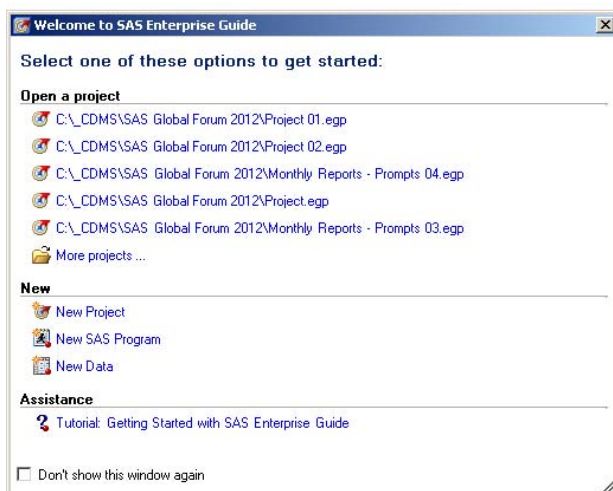
Early versions of SAS Enterprise Guide (EG) met with only lukewarm acceptance among many SAS programmers. As EG has matured, however, it has proven to be a powerful tool not only for end-users less familiar with SAS programming constructs, but also for experienced SAS programmers performing complex ad 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 perform 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?" However, some 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.

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 his or her SAS knowledge to the development of 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 the same as a SAS program; after all, a SAS program is little more than a collection of related PROC and DATA steps, LIBNAME statements, and comments. 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.

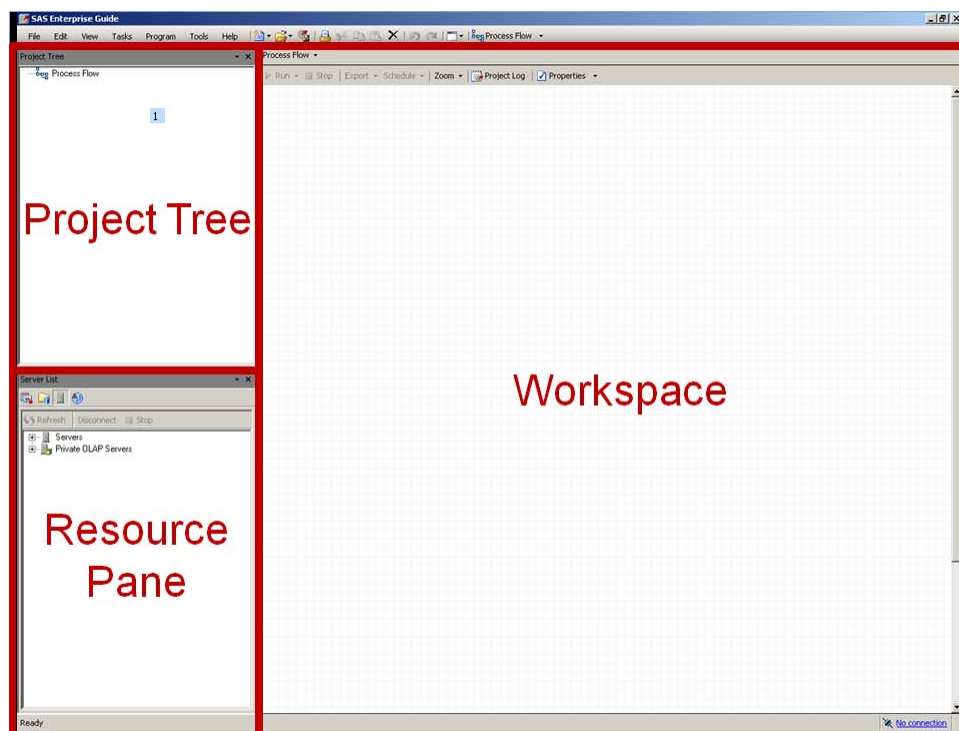


Take a Fresh Look at SAS® Enterprise Guide®: From point-and-click ad hocs to robust enterprise solutions, continued

So, in what ways does an EG project differ from a SAS program? First, not only can EG projects include links to external SAS programs, but, as is demonstrated later in the paper, they can contain **Embedded SAS Programs** which exist solely within the confines of the project—having no external .sas file. Further, EG contains built-in support for **Conditional Processing** to control the branching of code execution and user interface components for presenting users with **Prompts** for assignment of values that will drive code execution—for example, through the assignment of macro variable values in a PROC SQL WHERE clause. Beyond these superficial differences (which have more to do with differences in the interfaces than the functionality), 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 Display Manager 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, therefore, is to gain a basic understanding of the user interface in which they are built.

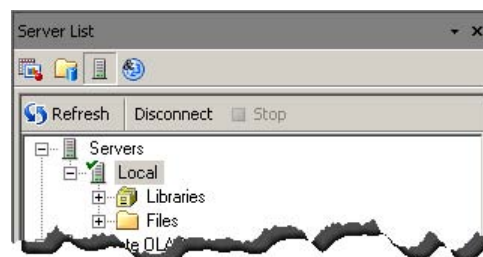
ENTERPRISE GUIDE INTERFACE

After launching Enterprise Guide and choosing "New Project" to navigate past the splash-screen¹, you are presented with a screen comprised of two docked windows (the **Resource Pane** and the **Project Tree**) and the **Workspace**—which technically is not a "window" since it cannot be closed, minimized, etc. independently from the application.



RESOURCE PANE. The Resource Pane 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.

SERVICES. Server resources refer 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) hints at the key characteristic describing EG; it is essentially a graphical user interface (GUI) to the SAS System.



¹Of 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.

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

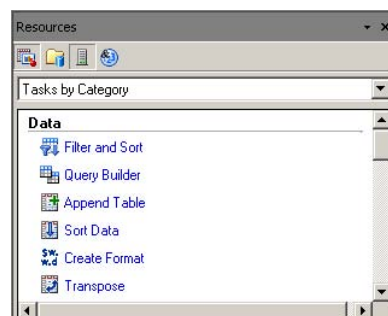
To confirm that EG is using your local SAS installation, start EG and navigate to the server "Local". After "Local" is started, launch SAS. You should notice that SAS gives you a warning that "User preferences will not be saved". This is because your local SAS environment is already in use—as the SAS engine for Enterprise Guide. Enterprise Guide generates SAS code necessary to execute the tasks you have specified and (when running in stand-alone mode) uses the local SAS engine to execute 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.

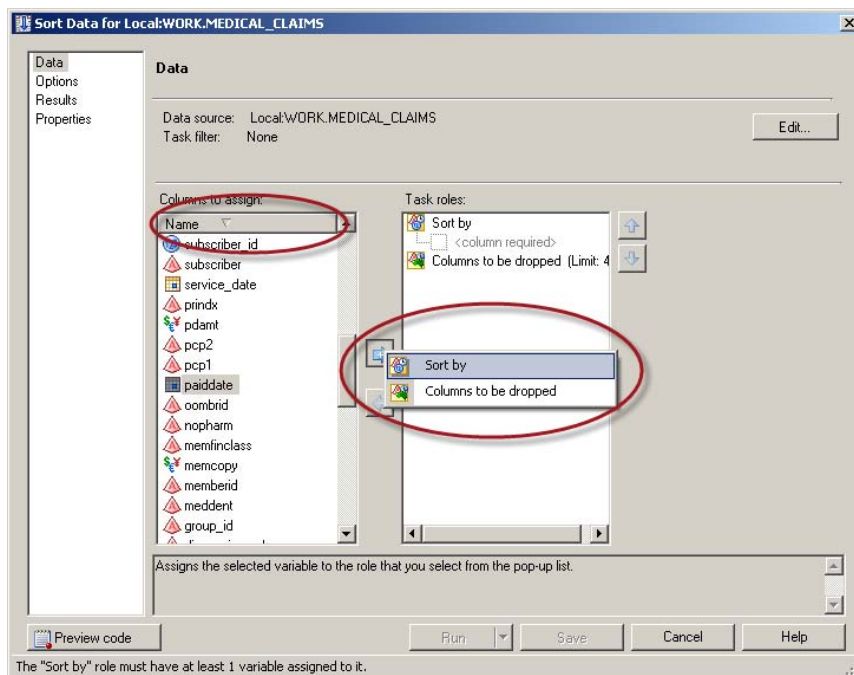


TASKS. The majority of work performed in an EG project is accomplished by adding tasks to the project. As seen in the figure to the right, some of the entries in the Task resource have names that are remarkably similar to the names of PROCs you might use in SAS. In fact, the "Sort Data" task sorts data using PROC SORT, for example. 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 applied to the execution of that task. This is where many experienced SAS programmers first throw up their hands in exasperation because they can very likely write SAS code for a given PROC faster than they can point and click their way through the corresponding Enterprise Guide 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 words "paid" and "date" in it, but you might not remember whether the variable is "paiddat", "paid_date", or "date_paid". In SAS you would either navigate to the Explorer, right-click on the dataset, 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.

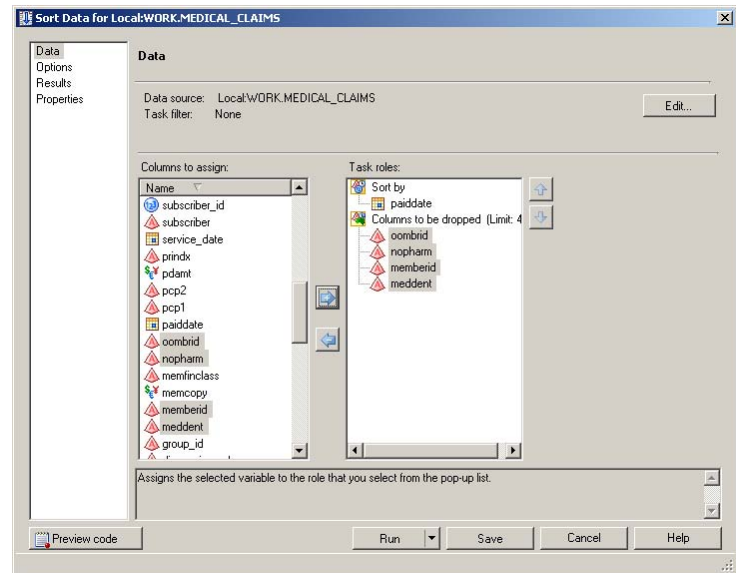
By contrast, in the Sort Data task, you can simply click the "Name" heading under "Columns to assign" to sort the column names in ascending or descending order, scroll down the list and click on the variable of interest—in this case "paiddat". Once the variable is found, however, you are confronted with 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 "paiddat" this variable should be assigned the "Sort by" role. As seen in the adjacent figure, once the variable of interest is selected, you can click the button to assign the variable to that role in the current task. Alternatively, you can drag "paiddat" to the "Sort by" role in the "Task roles" list.



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

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 dataset 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 accept 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>;
```



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

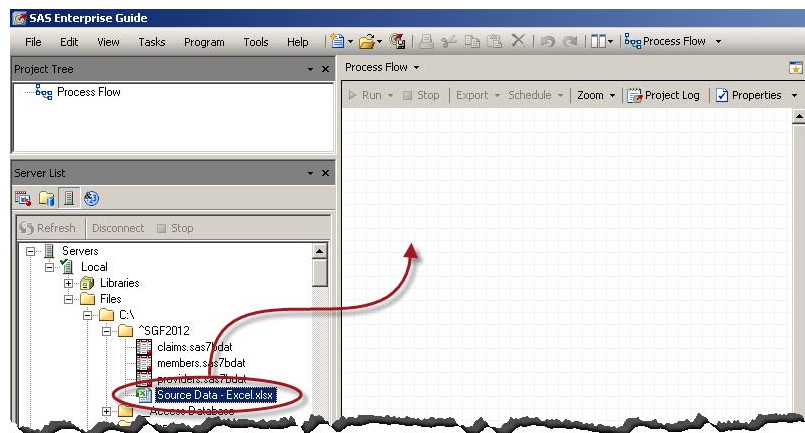
```
PROC SORT DATA=WORK.MEDICAL_CLAIMS
  OUT=WORK.SORTSortedMEDICAL_CLAIMS (
  LABEL="Sorted WORK.MEDICAL_CLAIMS"
  DROP= oombid nopharm memberid meddent);
  BY paiddat;
```

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 syntax that needs to be provided to SAS in order to perform 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.

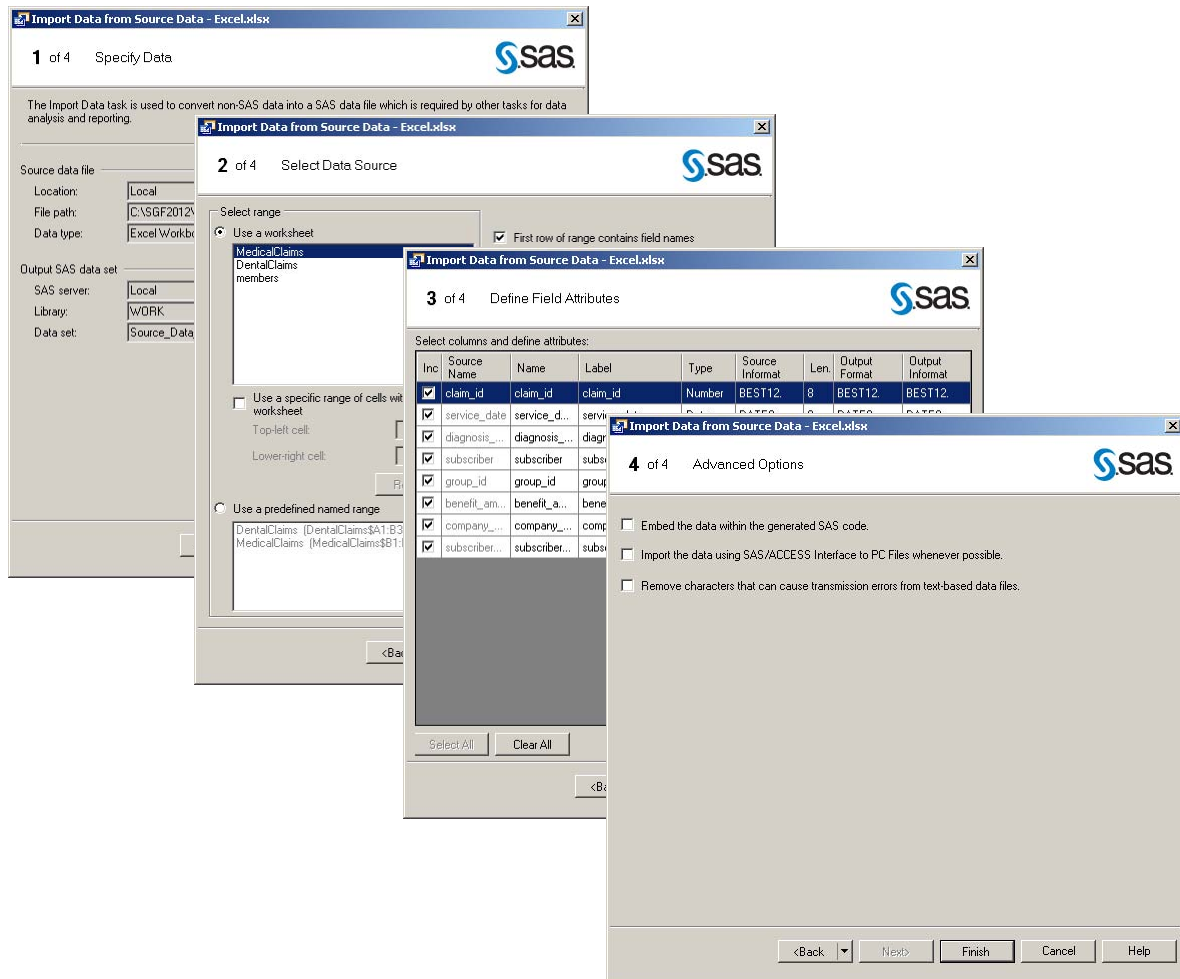
PROMPTS. Prompts play an important role in creating enterprise-level solutions for analysts and report writers. This resource provides a flexible way to present users with interface components with which they can indicate single or multiple values that will drive execution of a project. The specified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

WORKSPACE. The workspace contains **Process Flows** and **Document Windows**. The process flow in 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 onto a process flow.

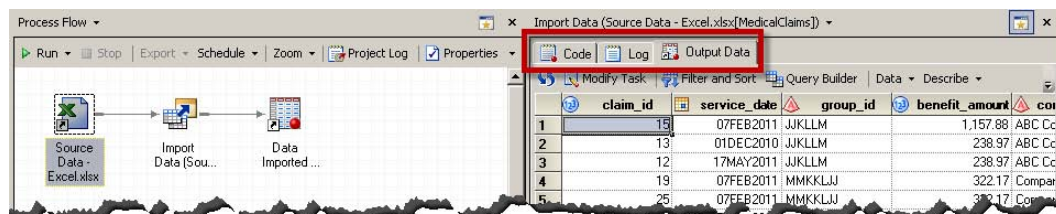


Take a Fresh Look at SAS® Enterprise Guide®: From point-and-click ad hocs to robust enterprise solutions, continued

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; in response, 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 current task.

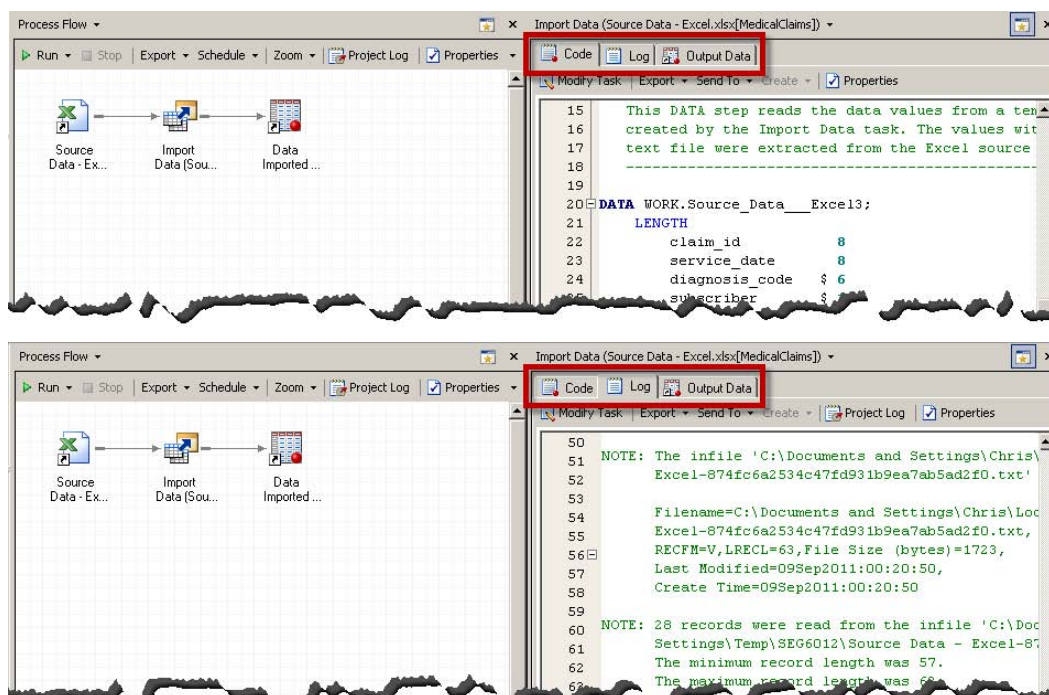


Once all of the steps involved in this task are specified, clicking Finish executes the task, opens the resulting SAS dataset in a **Document Window**, and generates, in the process flow, a graphical depiction of the relationship between the source dataset, the Import Data task, and the output dataset.



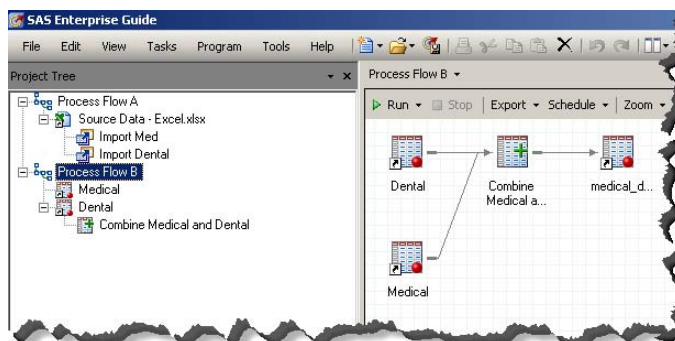
Note that the document window also contains tabs for the SAS code generated by EG as well as the log entries generated by the execution of that code.

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



To summarize the purpose of the workspace, it serves as a container for process flows that are used to organize tasks and document windows that are used to view output datasets generated by those tasks as well as the code and log entries associated with task execution.

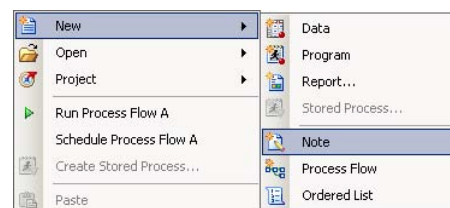
PROJECT TREE. The project tree serves to organize the project by providing quick reference to the lineage of datasets, the tasks that use and generate those datasets, and the process flows used to organize the tasks in logically meaningful groups within the project. In the adjacent project tree, we see 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", and you can see from the depiction of "Process Flow B" in the workspace, that these datasets are both inputs to the task "Combine Medical and Dental". Through careful naming of tasks and datasets and the logical grouping of tasks within process flows, the project tree can be a useful tool for quickly navigating a complex project.



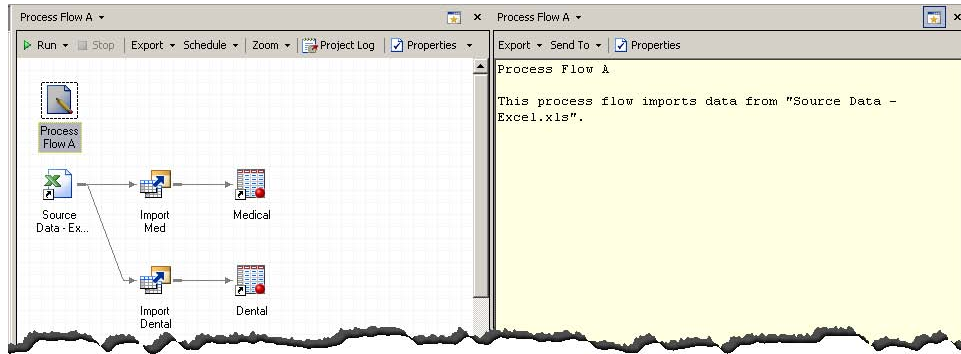
WORKING WITH TASKS AND PROCESS FLOWS

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.

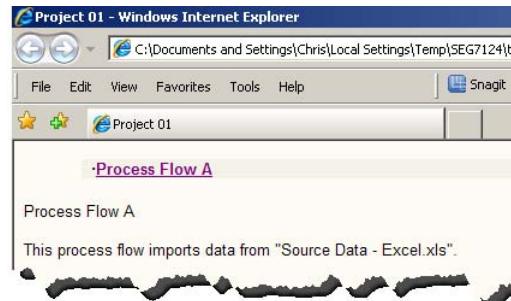
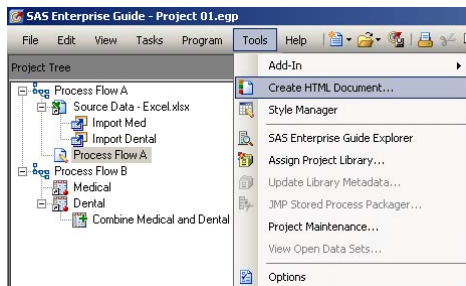
PROJECT COMMENTING. Although some Enterprise Guide users may have sophisticated source code repository systems, others may rely on file naming conventions and comments within their code to track changes. For this latter group of users, one of the nice features of EG is that you can add **Notes** to your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



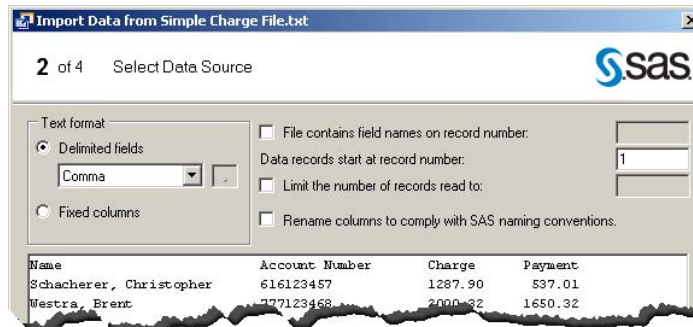
Take a Fresh Look at SAS® Enterprise Guide®: From point-and-click ad hocs to robust enterprise solutions, continued



The note can then be used as an input to the "Create HTML Document" option under the "Tools" menu to generate an HTML document that combines all of the notes in your project file².



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 triggers (by default) the creation of an Import Data 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 Data 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 delimiter.

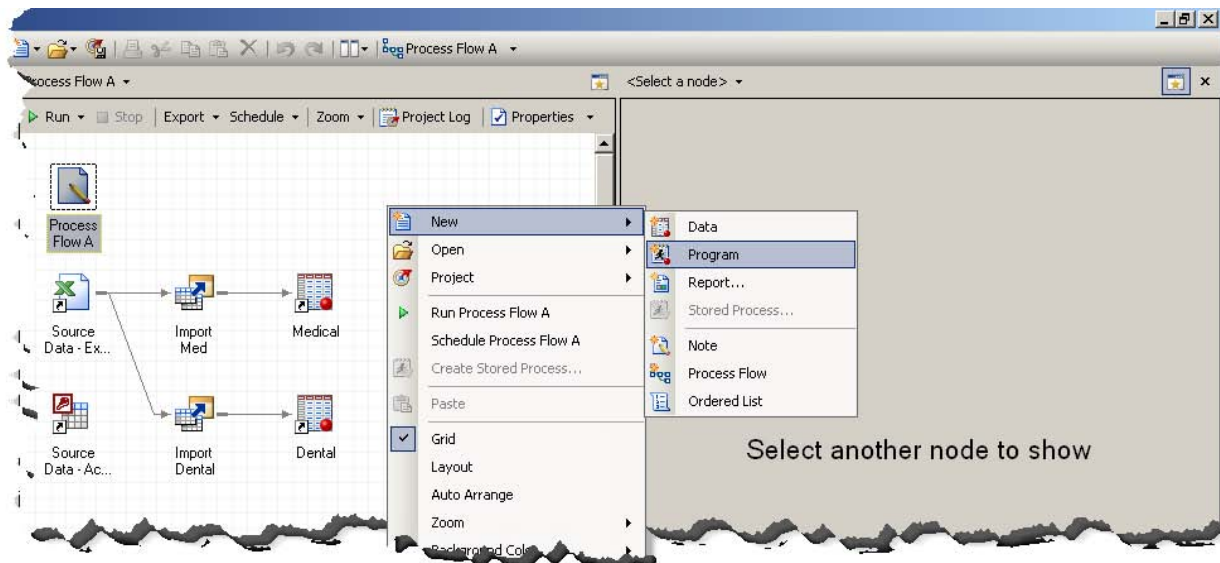


The ability to drag and drop data files into a project provides the EG user with a great advantage compared to having to remember the syntax of INFILE statements, PROC IMPORT, and the like for every file type and configuration they might encounter. To be sure, there are files (especially files with multiline 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.

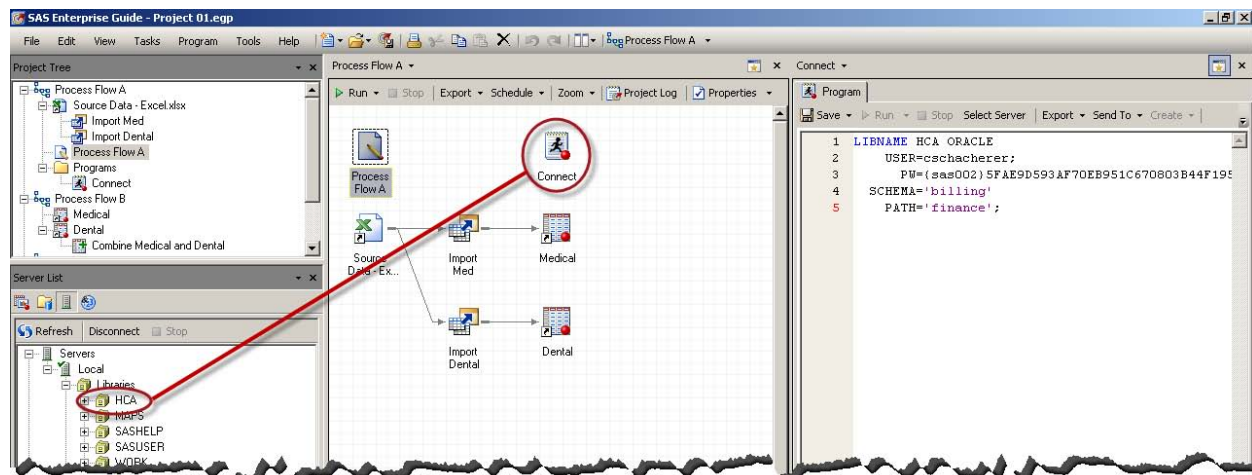
For data that resides in relational databases such as Oracle®, SQL Server®, DB2®, etc., a different approach is required for the Enterprise Guide user connecting only to their local installation of SAS. Because the tasks in Enterprise Guide are built for utilizing SAS to perform analyses and data manipulations (and not for configuring your SAS session), creating a LIBNAME 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.

² The reader is also referred to Hallahan & Atkinson (2006) for examples of adding analytic output to HTML documents using the "Create HTML Document" option.

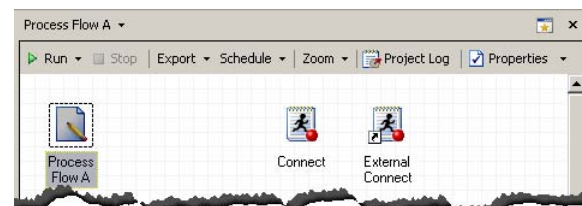
Take a Fresh Look at SAS® Enterprise Guide®: From point-and-click ad hocs to robust enterprise solutions, continued



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 ("Connect"), a refreshed view of the Local server shows that the project now has access to the HCA library³. 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.



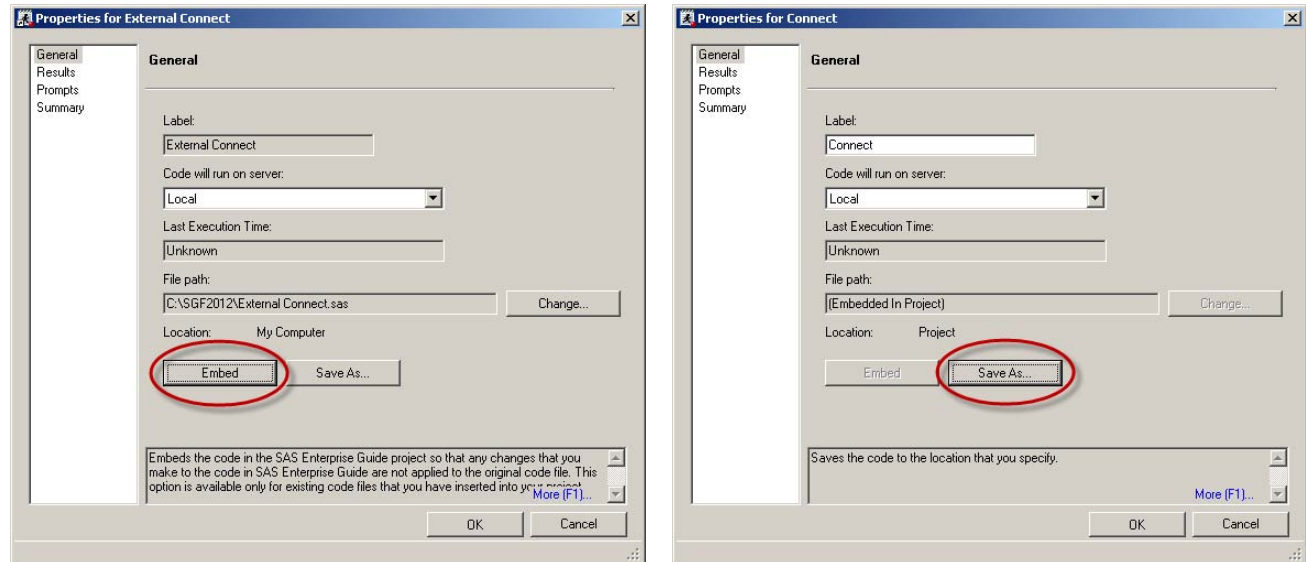
LINKING & EMBEDDING SAS PROGRAMS. In the previous example, the SAS program "Connect" was created within the project as an **Embedded Program**. As such, it exists only within the EG project where it was created; there is no ".sas" file saved externally. If, on the other hand, we had included the existing external SAS program "External Connect.sas" 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 case of the embedded program, changes can only be made by editing it in the Enterprise Guide project. The linked program, on the other hand, can be altered without opening the Enterprise Guide project containing the link to the program.



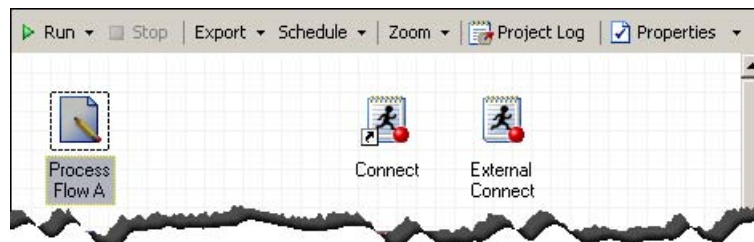
³ For information on using SAS/Access to define libraries see Levine (2001) and SAS (2004a, 2004b).

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

You might decide later that an embedded program like "Connect" might be useful in several of your EG projects because you often connect to the same database. In that case, you might want to make it an external SAS program so that you can manage it 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.



The result of the preceding two operations is that "Connect" is now a linked program and "External Connect" is now embedded in the project.

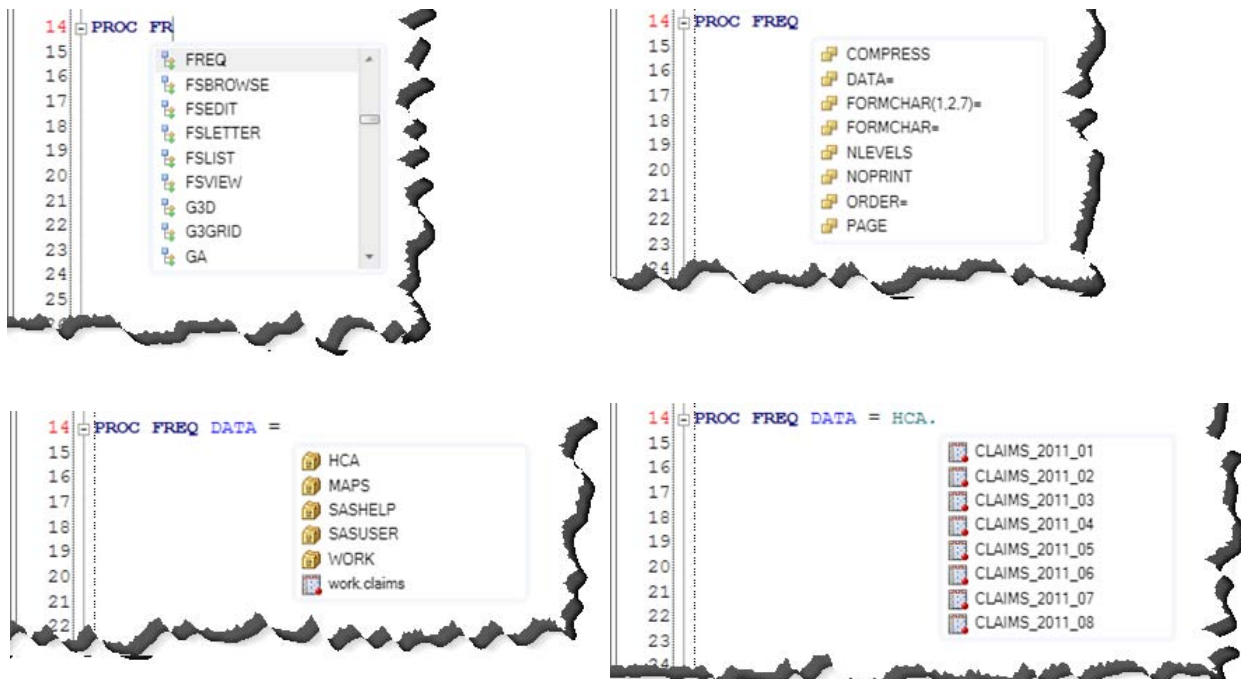


In Enterprise Guide 4.3, once you have written the code to establish libraries that point to your relational database management system, you can move the program(s) used to make these connections to a new process flow, name that process flow "Autoexec", and the code will be executed automatically each 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 Enhanced Editor used to write and edit them in Enterprise Guide significantly improves upon the Enhanced Editor available in the SAS Display Manager. As described in detail elsewhere (Bangi, Hemedinger, and Slocum [2010]; Fecht and Dhillon [2011]; Ravenna [2011]), the version of the Enhanced Editor that is available in EG 4.3 provides a number of tools developed specifically for the SAS programmer. Many common frustrations experienced 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.

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

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.



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.

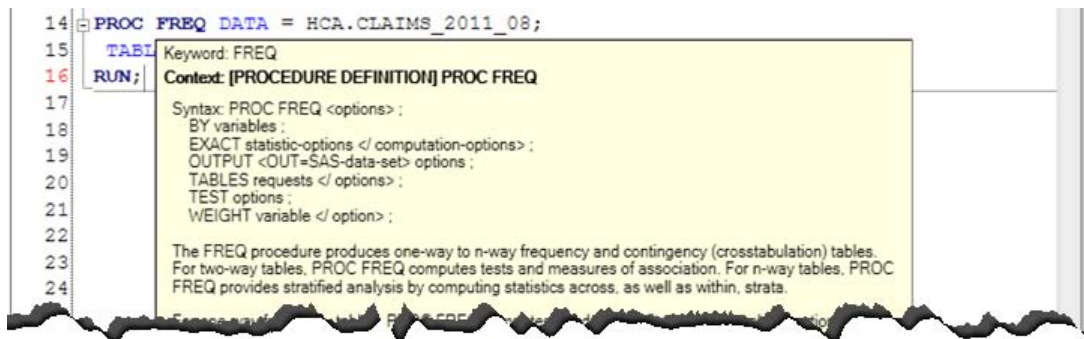


Beyond these enhancements to facilitate the mechanics of SAS programming, the EG 4.3 Enhanced Editor also provides programming support in the form of **Integrated Syntax Help** and **Function Completion**. Function completion presents users with the syntax alternatives available for a given function,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



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

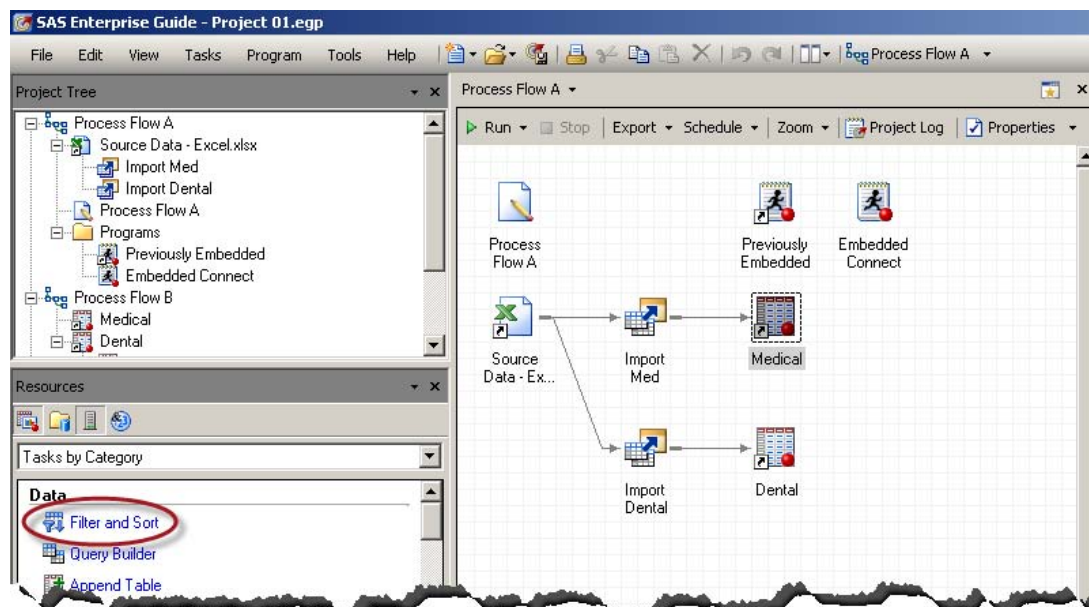
Similarly, integrated syntax help provides context-specific help topics related to the keywords being typed in a PROC or DATA step or OPTIONS statement. By simply hovering over the keyword, the user is presented with help information related to that keyword.



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 (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." (Bangli, Hemedinger, and Slocum, 2010, p. 10.). Even if you choose to ignore 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 the new enhancements to the Enhanced Editor.

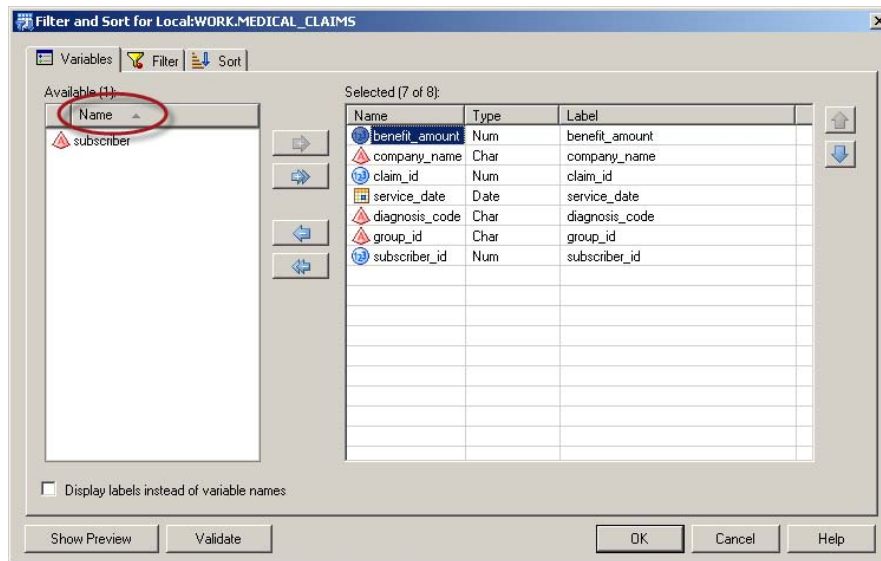
Whether accessing the data for your project is done by dragging and dropping files or by referencing data in SAS libraries, one of the fundamental activities for which you will be using EG is preparing data for analysis and reporting. This can include everything from sorting data for analysis across 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 date of service and claim ID.




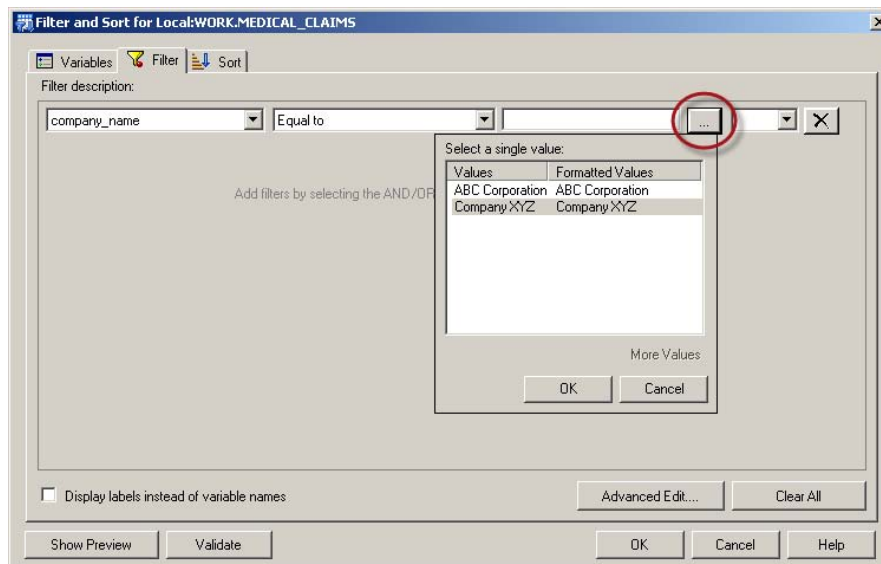
Like many of the tasks in EG, the user interface for specifying the Filter and Sort task has several useful features. First, when the task is added to the process flow (by either clicking it in the Tasks resource or by choosing it from the Tasks menu), the currently selected (or most recently selected)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

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

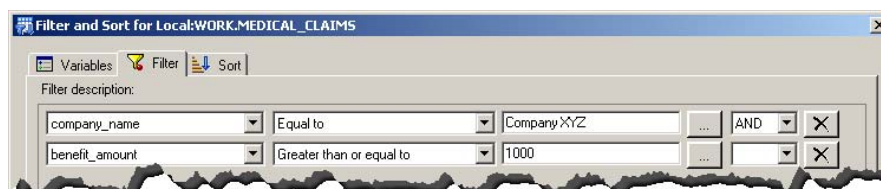
The task also affords the ability to alphabetically sort the names of the variables, which aids in selection of the variables to be included in the resulting dataset. Variables can be selected for inclusion in the resulting dataset by selecting one or more variables and clicking the single arrow , dragging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



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.

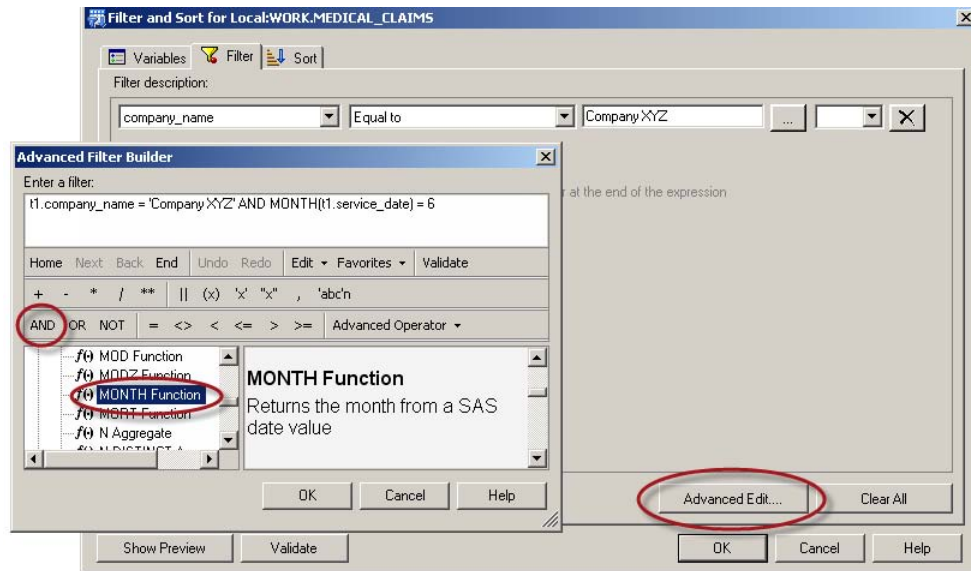


To add a second filter criterion, add an And/Or operator to the filter and enter the next filter criterion.

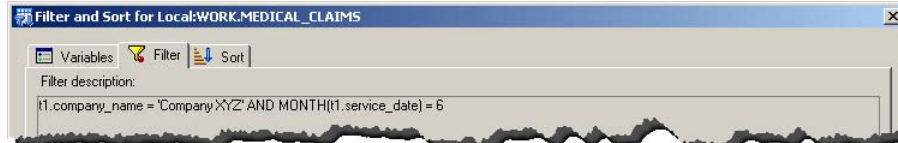


Take a Fresh Look at SAS® Enterprise Guide®: From point-and-click ad hocs to robust enterprise solutions, continued

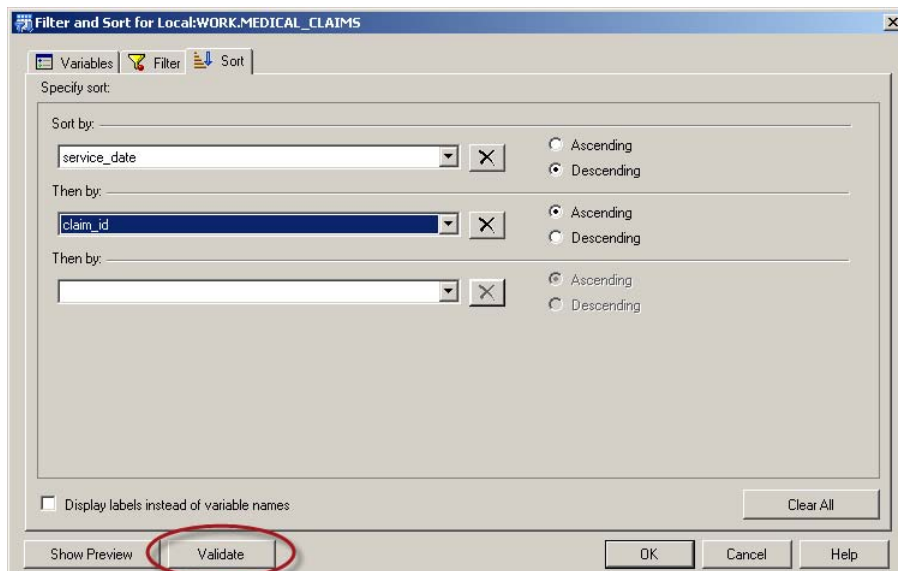
To build more complex criteria involving SAS functions, algebraic expressions, or advanced operators, click on the Advanced Edit button to navigate to the Advanced Filter Builder. In the following example, an "AND" operator is added to the set of conditions that define the filter and the MONTH function is used to specify that the filter should also include a restriction to only select those records where the date of service was 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.



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

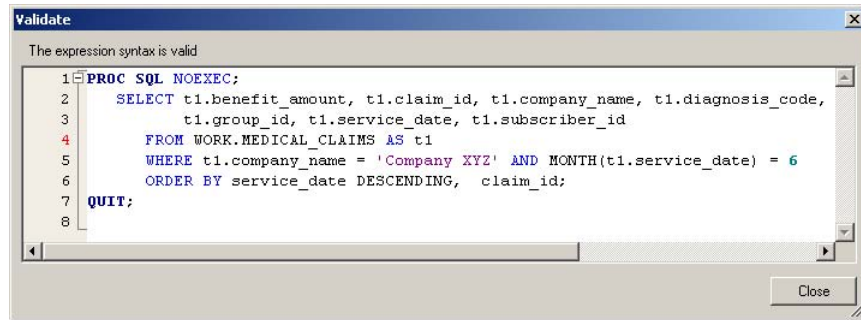


After selecting the variables for inclusion and building your filter logic, the Sort tab can be used to order the records in the resulting dataset. In the current example, the dataset will be sorted in descending order of the values of "service_date" and then in ascending order of the values of "claim_id".

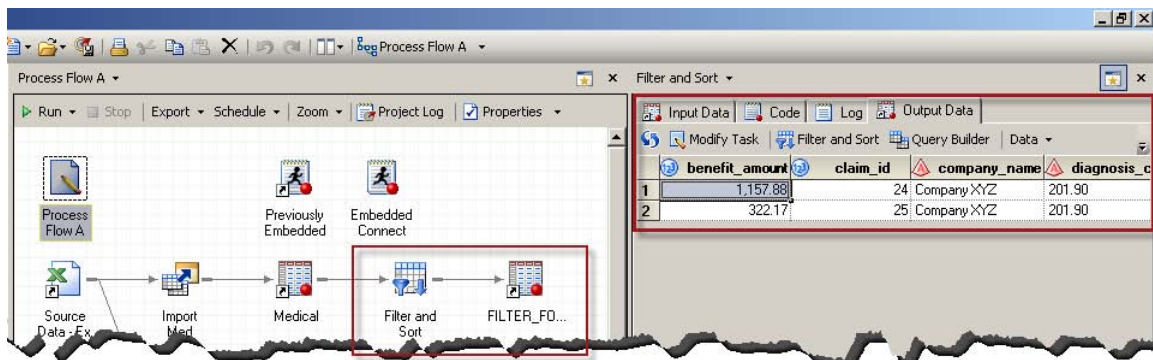


Take a Fresh Look at SAS® Enterprise Guide®: From point-and-click ad hoc to robust enterprise solutions, continued

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:



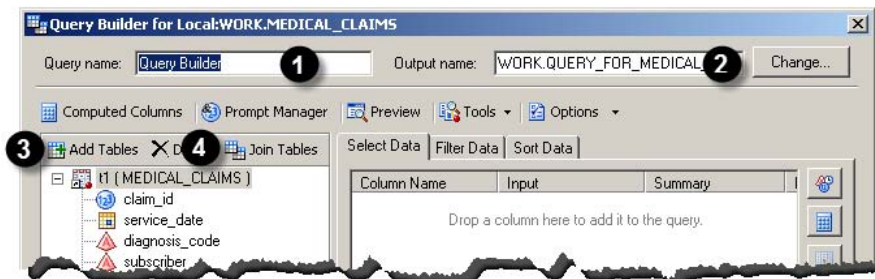
After clicking Close on the Validate popup, you are returned to the task interface, where clicking OK executes the task—adding the task to the process flow and generating the output dataset.



As described earlier, Enterprise Guide tasks provide a graphical tool for building SAS code associated with PROC and DATA steps. The Validate window presented in the preceding example serves to reinforce this point; the result of the point-and-click specification of the Filter and Sort task was the generation of SAS code capable of filtering and sorting the source dataset just as we intended. Before the experienced programmer dismisses this as a "cheat", consider 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 your changing needs is definitely worth the minimal time necessary to master the tasks—especially when you already know the PROCs on which they are based.

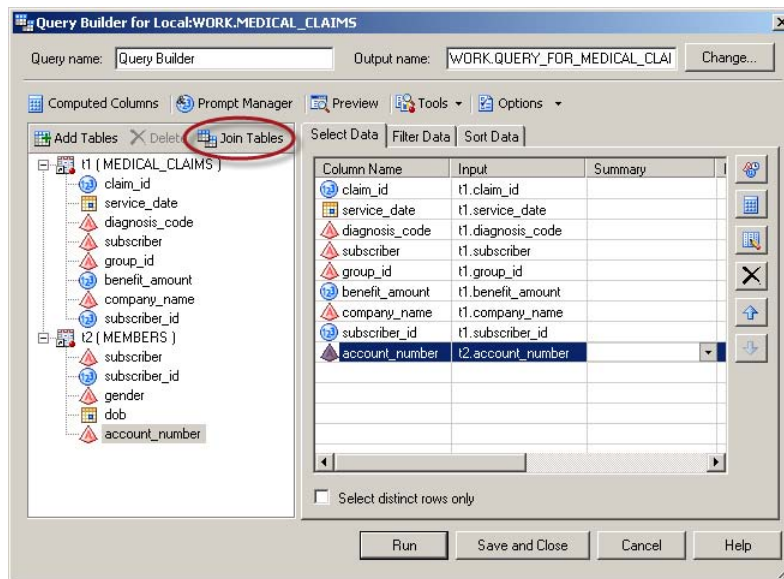
THE QUERY BUILDER TASK. One of the most important EG tasks for both developers and end-users to master is the Query Builder task—which can be used to subset, join, and summarize datasets. As with the Filter and Sort task, the Query Builder task defaults to inclusion of the dataset that was most recently selected or created in the current process flow.

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 output dataset, and (4) specify the join condition that will be used to associate records from each of the included tables.

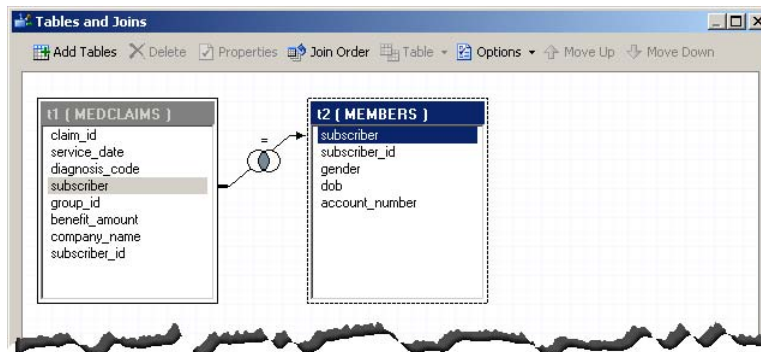


Take a Fresh Look at SAS® Enterprise Guide®: From point-and-click ad hocs to robust enterprise solutions, continued

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 for inclusion in the new dataset by dragging and dropping it onto the Select Data tab.

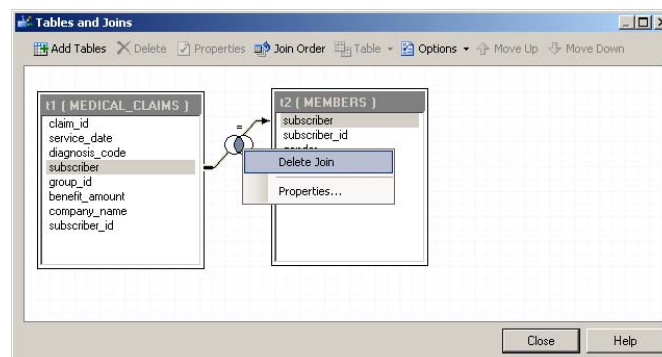


To specify how the tables should be joined, click on Join Tables to bring up the Tables and Joins window.



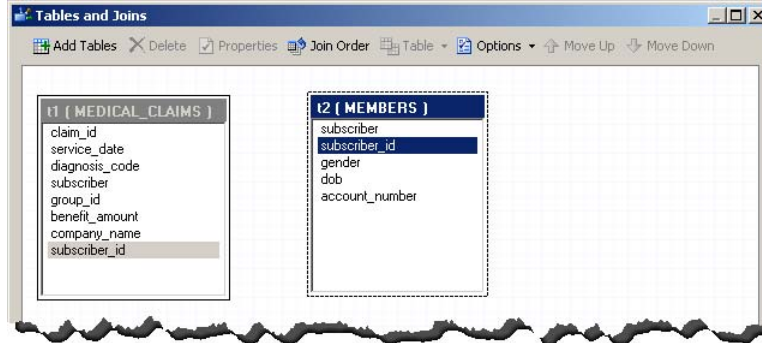
By default, Query Builder will attempt to determine which fields should be used to join the tables included in the query. In this case Query Builder has detected that both datasets contain the variable "subscriber" and assumes that you want to perform an equi-join between the members and their healthcare claims based on this variable. 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) 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.

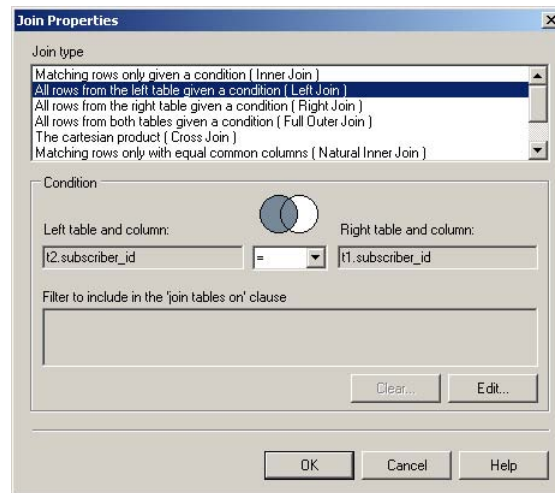


Take a Fresh Look at SAS® Enterprise Guide®: From point-and-click ad hocs to robust enterprise solutions, continued

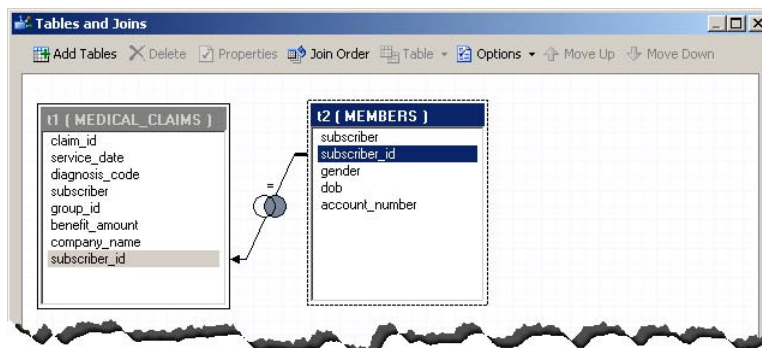
Then the new join is created by dragging "subscriber_id" from the "members" table onto "subscriber_id" on the "medical_claims" table.



Dragging the field 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".

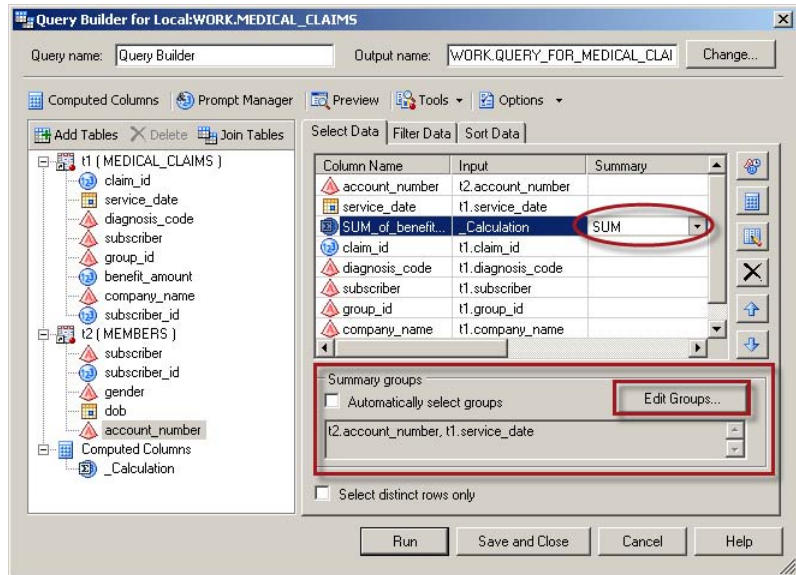


Once the new join type is selected, click OK to assign the new join condition and close the Tables and Joins window to return to the Query Builder's main interface.



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

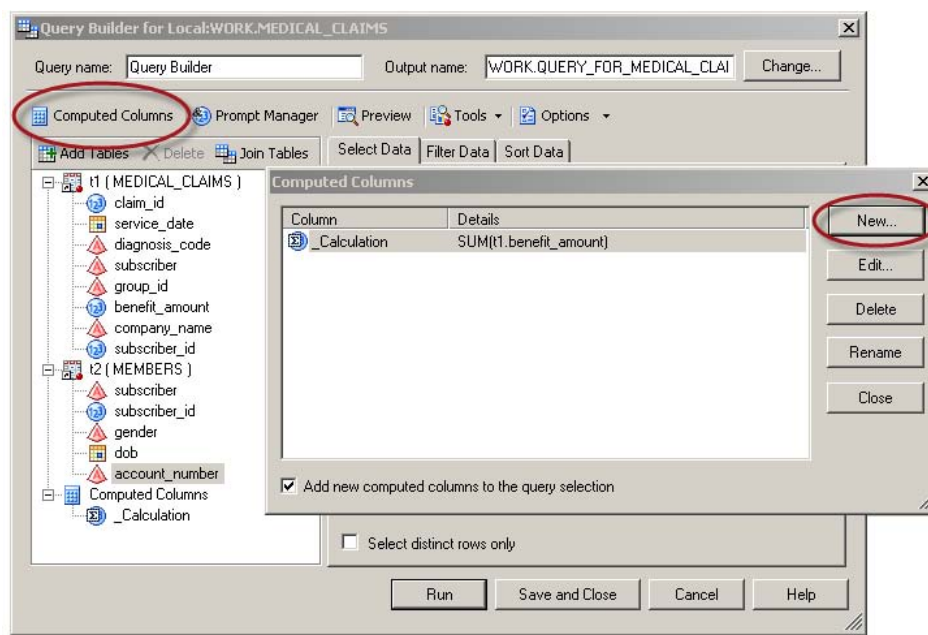
Upon returning to the Query Builder's main window, you can rearrange the order of the variables in the Select Data tab using the up and down arrows to move the column names up and down in the presentation order (note the order of "t2.account_number" and "t1.service_date"). Additionally, you can add summary 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, however, when a summary function is specified, the checkbox labeled "Automatically select groups" is checked for you, and all other selected variables are added to 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 your grouping variables.



To change the sort order (ORDER BY clause) or to add a filter condition (to the WHERE clause), the Filter Data and Sort Data tabs provide the same highly intuitive interfaces demonstrated in the earlier example of the Filter and Sort task.

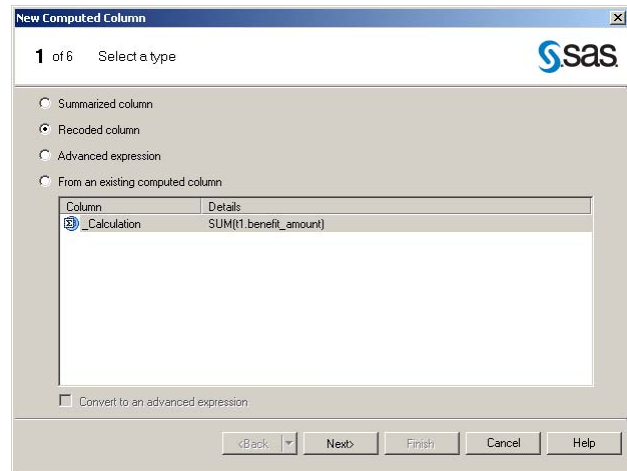
In addition to joining, summarizing, sorting, and filtering your data, you often need to create new variables or transform existing variables when creating a new dataset. In PROC SQL, you might accomplish this using a CASE statement to perform recoding based on conditional logic, by writing an arithmetic expression, or through the application of SAS functions. In the Query Builder task all such transformations are achieved using Computed Columns.

In order to create a computed column, click the Computed Columns button on the Query Builder task's main screen and click New in the Computed Columns window.

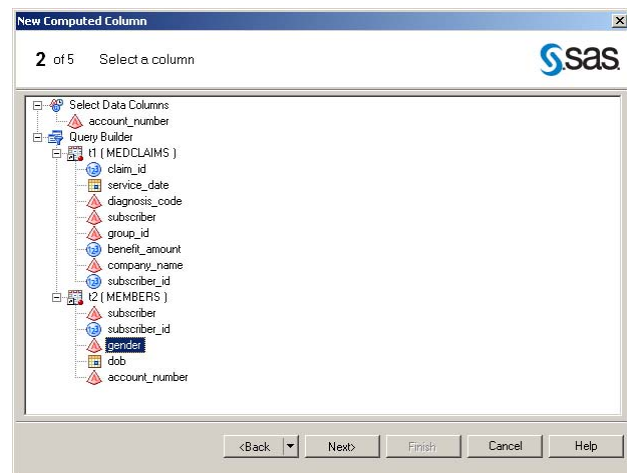


Take a Fresh Look at SAS® Enterprise Guide®: From point-and-click ad hocs to robust enterprise solutions, continued

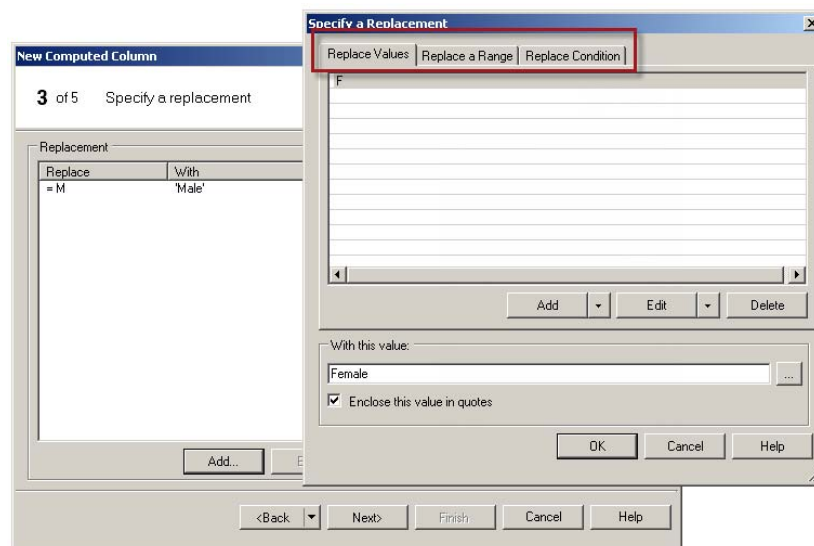
Next, select the computed column type. Summarized columns are those that are computed using SQL aggregate functions. Recoded columns are those that assign values based on some logical condition evaluated for each record. Advanced expression columns are those created using the Advanced Expression Builder (similar to the Advanced Filter Builder in the previous Filter and Sort example), and columns produced "From an existing computed column" are those for which an existing computed column is used as the basis for the computation that produces the new column. In the current example, the values "M" and "F" in the "gender" variable of the "members" dataset will be recoded to "Male" and "Female", respectively, in the new variable "member_gender".



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



The values to be recoded are then specified along with their replacement values. Note that a number of replacement strategies are supported. One can specify individual values to be replaced (e.g., "F" recoded as "Female"), 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").



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

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 new 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. Change...

<Back Next Finish Cancel Help

Finally, in Step 5, you are presented with a summary of the properties for the new computed variable. At this point, you can either click Finish to complete the specification of the new variable and return to the Query Builder screen or click Back to step back through the New Computed Column wizard and alter the specifications for the new variable.

New Computed Column

5 of 5 Summary of properties

Alias: Calculation

Column: member_gender

Type: Character

Format: \$CHAR10

Length: Default

Summary: None

Expression:

```
CASE
  WHEN 'F' = t2.gender THEN 'Female'
  WHEN 'M' = t2.gender THEN 'Male'
  ELSE t2.gender
END
```

<Back Next Finish Cancel Help

As in the previous Filter and Sort example, you can preview the SAS code that will be generated by the Query Builder task before running it. Click Preview on the Query Builder task window and the 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 Change...

Computed Columns Prompt Manager Preview Tools Options

Add Tables Delete Join Task

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

Computed Columns

- Calculation
- member_gender

Preview

Code Results Log

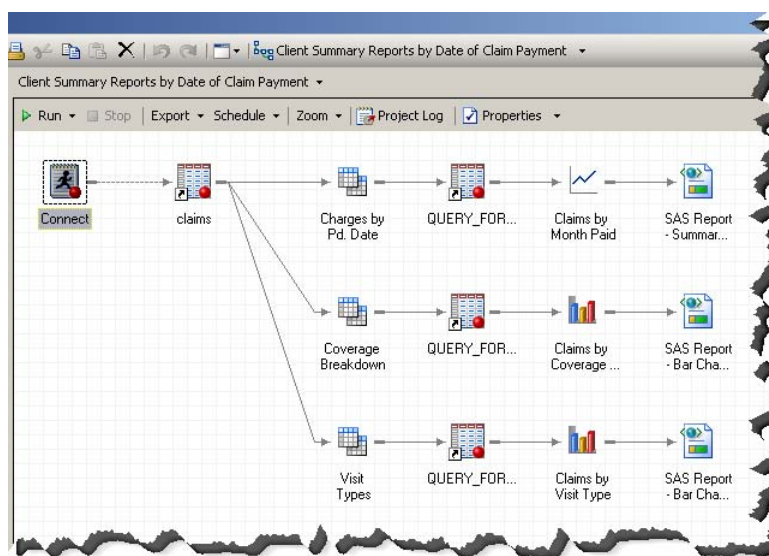
```
1 % eg_conditional_drops(WORK.QUERY_FOR_MEDICAL_CLAIMS);
2
3 PROC SQL;
4 CREATE TABLE WORK.QUERY_FOR_MEDICAL_CLAIMS AS
5 SELECT t2.account_number,
6 t1.service_date,
7 /* SUM of benefit_amount */
8 (SUM(t1.benefit_amount)) FORMAT=COMMA12.2 AS SUM_of_benefit_
9 t1.claim_id,
10 t1.diagnosis_code,
11 t1.subscriber,
12 t1.group_id,
13 t1.company_name,
14 t1.subscriber_id,
15 /* Calculation */
16 (CASE
17 WHEN 'F' = t2.gender THEN 'Female'
18 WHEN 'M' = t2.gender THEN 'Male'
19 ELSE t2.gender
20 END) FORMAT=$CHAR8. AS Calculation
21 FROM WORK.MEMBERS AS t2 LEFT JOIN WORK.MEDICAL_CLAIMS AS t1 ON (t2
22 GROUP BY t2.account_number, t1.service_date;
23 QUIT;
```

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

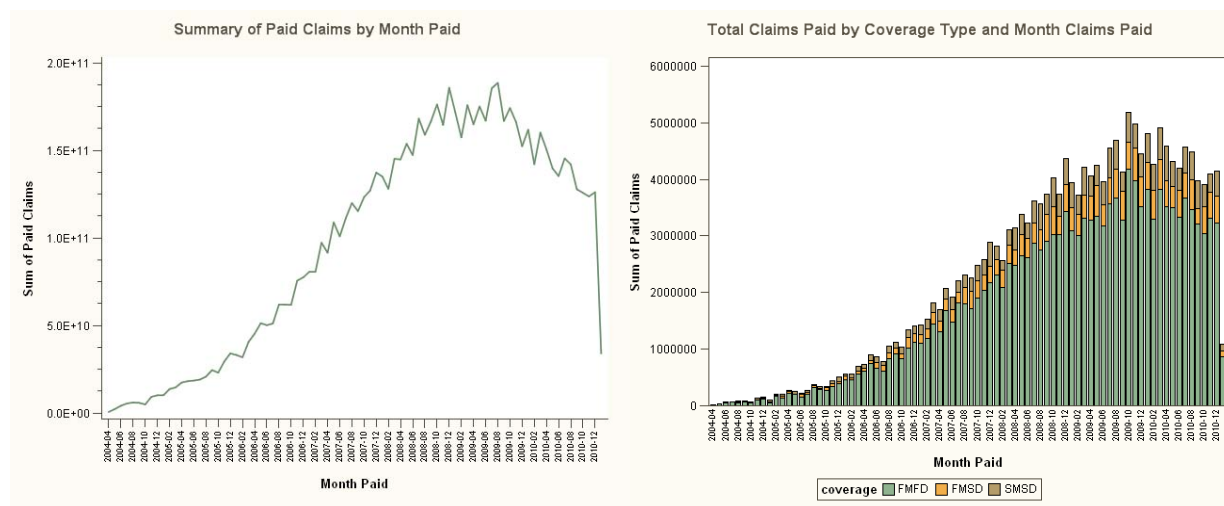
As the Query Builder and Filter and Sort task examples demonstrate, one of the main goals of EG is to enable end-users without SAS programming expertise to utilize the power of the SAS programming environment to manipulate and analyze data. One should not conclude, however, that EG obviates the need 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 a 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 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 Builder tasks followed by Line Graph and Bar Chart tasks.

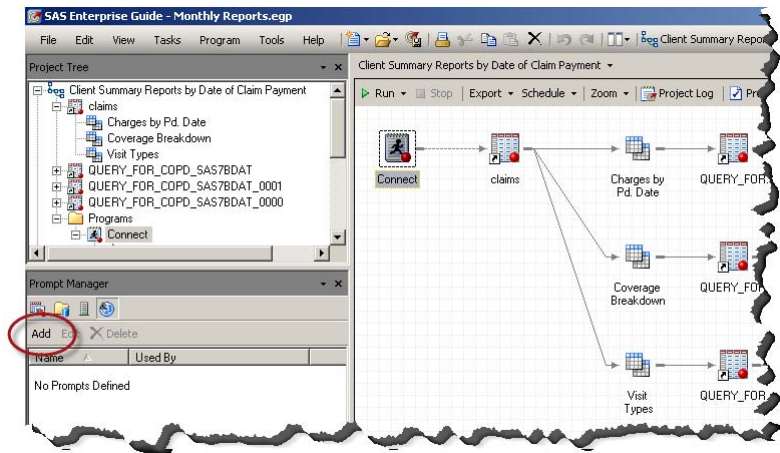


The resulting graphs are exactly what she wanted, and after she shares them with a colleague, he decides that he would like similar graphs delivered monthly for his clients. You suspect that requests for these graphs could grow rapidly as they get shared with other account managers. Wanting to meet the needs of these data consumers, you rethink the original project and realize that you need to build the queries with the flexibility to change the client company with each execution of the project.



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

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 "Monthly Reports" project will be run.



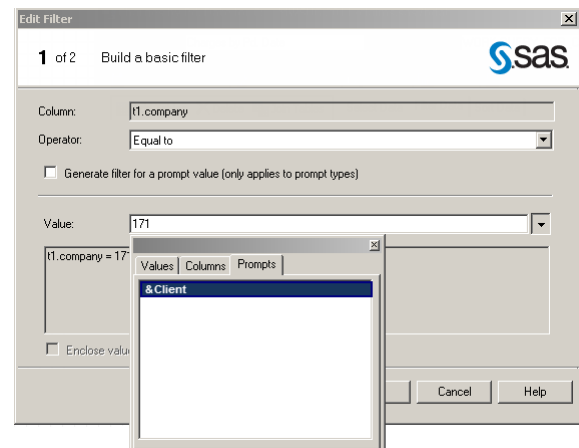
Because the "company_id" variable used in the WHERE clause of the queries is numeric, the Prompt Type for "Client" is specified as numeric. We want users of this prompt to select their client company by choosing it from a drop-down list, so "User selects values from a static list" is selected under "Method for populating prompt". As indicated in "Number of Values", users of this prompt will be allowed to select a single value from the list presented to them—the individual client for which the reports will be generated. 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 data source using the Get Values button. Finally, when the list is displayed to the end-user, the Formatted Value of the list entry will be appended with the Unformatted Value (e.g., "Company ABC [140]")—as this extra bit of information can be useful to users who are as familiar with the client code as they are with the Formatted (Displayed) Value. Once the prompt is built, click "OK" to save the prompt to the 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 is created by clicking Add in the Prompt Manager resource and giving the new prompt a name, instructive text to displayed in the prompt, and a description of the prompt. As shown in the prompt properties, this new "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.

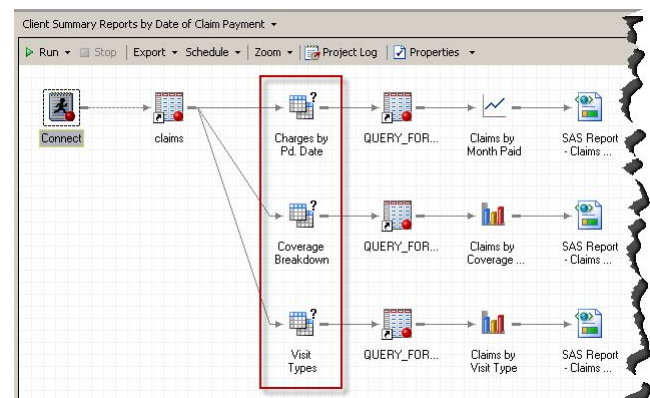
Unformatted Value	Formatted (Displayed) Value	Default
140	Company ABC	<input type="radio"/>
171	Company XYZ	<input type="radio"/>
182	JKL Corporation	<input type="radio"/>
212	MNO, LLC	<input type="radio"/>
255	Zerillion, LLP	<input type="radio"/>

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

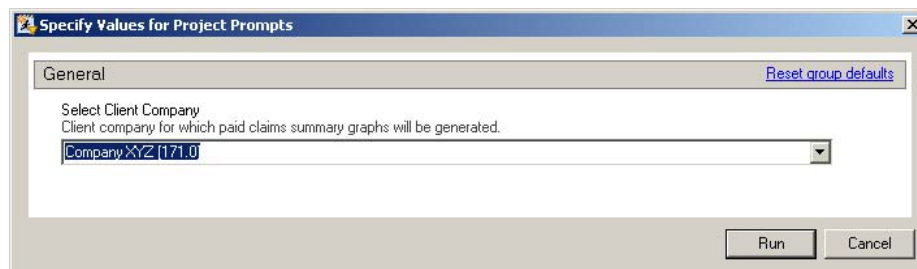
Next, the prompt needs to be associated with the Filter (or WHERE clause) of the Query Builder task. This association is made by selecting the Query Builder task in the process flow, right-clicking it, and choosing Modify <name of task>. In the adjacent figure, the filter of the "Charges by Pd. Date" Query Builder task is being edited. Instead of providing a specific company id (e.g., "171") as the value for this condition (as was done originally), navigate to the Prompts tab of the Value drop-down and select "&Client" to assign the prompt value as the evaluation criterion.



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 to indicate that the task now involves a prompt.]



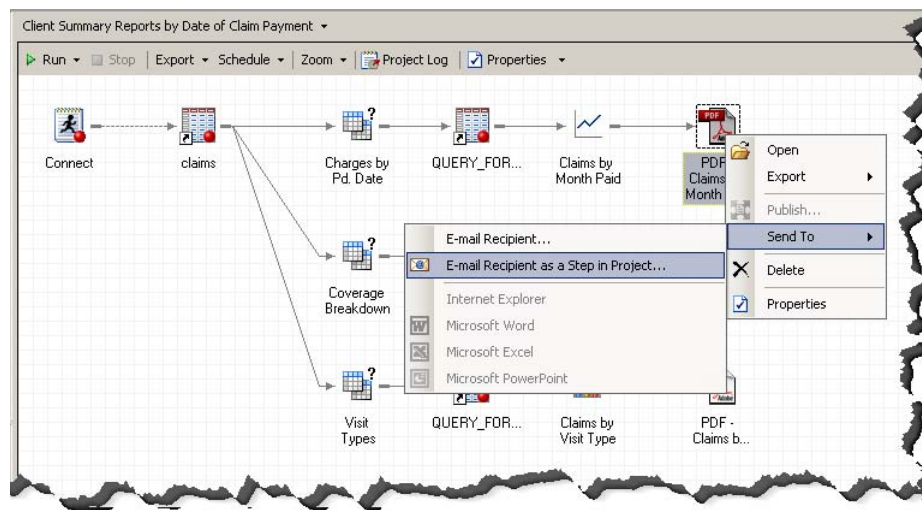
When the process flow is next run, the user will be presented with the new prompt.



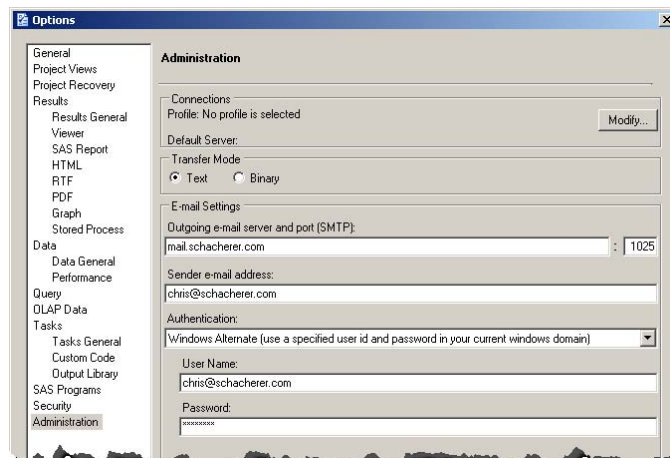
Once a prompt value is chosen by the user, clicking Run will execute the query builder tasks and produce the reports for the selected client company. To produce the graphs for a different company, all one needs to do is run the process flow again and choose a different value in the "Client" prompt.

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

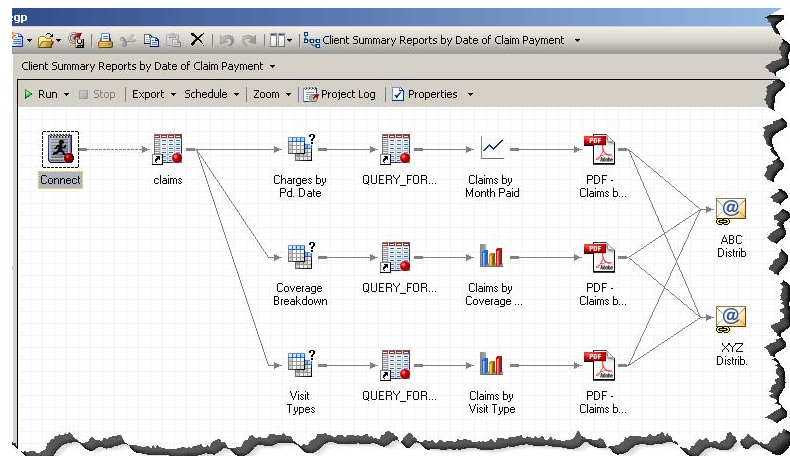
DELIVERY AUTOMATION. In addition to facilitating the flexible production of reports, SAS Enterprise Guide also has built-in functionality to facilitate the distribution of analytic output. To expand on the previous example, once the reports are generated, you could use the **Send To** functionality to 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, 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 is simply another task to be executed in the process flow.

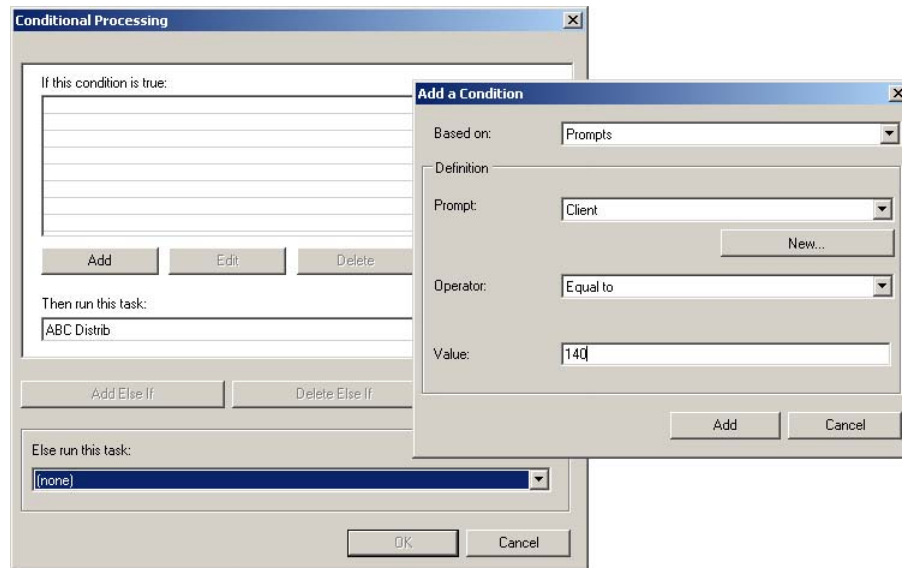


CONDITIONAL PROCESSING. One remaining challenge for the distribution of these reports is that for each client company there will likely be different distribution lists. For example, Mary might want to receive all Company XYZ reports, but not Company ABC reports, and the converse might be true for Bill. 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 a given 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**.

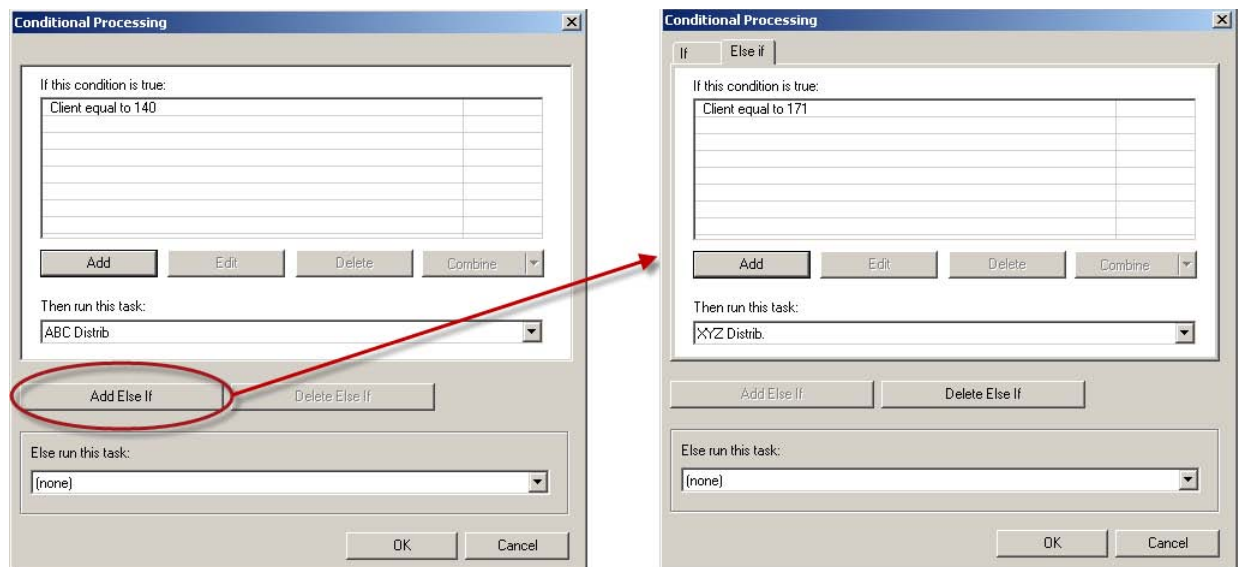


Take a Fresh Look at SAS® Enterprise Guide®: From point-and-click ad hocs to robust enterprise solutions, continued

In specifying the properties of this condition we indicate that it 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.

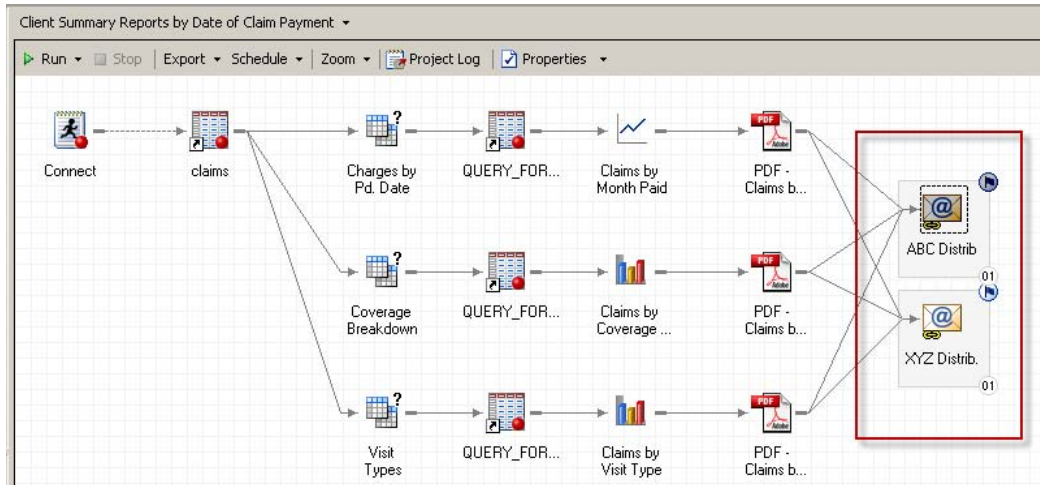


Once that condition is created, you can add an "Else If" condition to assess whether "XYZ Distrib" should be executed in those cases where the first condition "Client = 140" does not evaluate to "TRUE". If the value of "Client" is "171" instead of "140", "XYZ Distrib" will be run—sending the results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

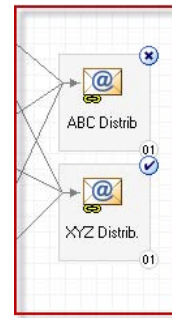


Take a Fresh Look at SAS® Enterprise Guide®: From point-and-click ad hocs to robust enterprise solutions, continued

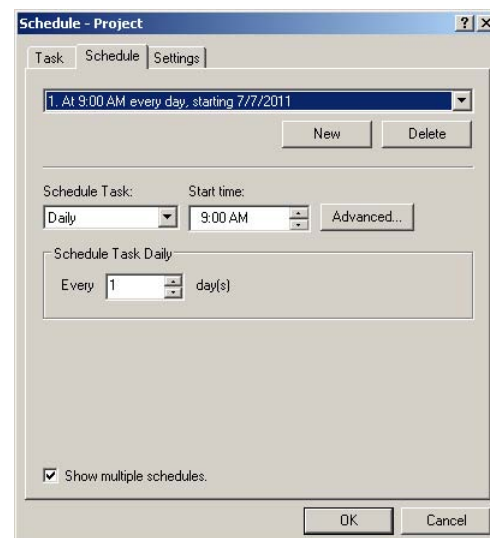
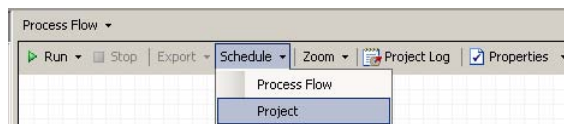
Once saved, conditional logic is denoted in the process flow by the addition flags to the "ABC Distrib" and "XYZ Distrib" icons—indicating that these tasks are contingent on the outcome of the new condition.



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 distribution 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 VB script that is run by the Windows Scheduler according to the designated schedule, and this script, in turn, launches the scheduled project or process flow via Enterprise Guide.



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

CONCLUSION

The inclusion of developer-centered functionality like prompts, conditional logic, and automated scheduling and delivery, 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 should not be dismissed as simply a point-and-click tool for end-users unfamiliar with SAS programming. At the same time, 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 colleagues 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.
- Hallahan, C. & Atkinson, L. (2006). Introduction to SAS® Enterprise Guide® 4.1 for Statistical Analysis. Proceedings of the 31st Annual SAS Users Group International Meeting. Cary, NC: SAS Institute, Inc.
- 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.
- 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
 E-mail: CSchacherer@cdms-llc.com
 Web: www.cdms-llc.com

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

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