

SAS ETL Studio: An Introduction to the Version 9 Data Warehousing solution

Kevin Davidson · FSD Data Services, Inc.

ABSTRACT

SAS ETL Studio was introduced in Version 9 and replaces SAS/Warehouse Administrator. This paper will provide a broad overview of the software with an emphasis on how the Extract-Transform-Load (ETL) features work. A sample 'job' will be built to demonstrate the capabilities and an overview of how ETL Studio communicates with SAS sessions to perform the job processing will be discussed.

INTRODUCTION

SAS ETL Studio enables you to manage ETL Process flows. What make up these process flows? Through its metadata, SAS ETL Studio provides a single point of control for managing the following items:

- Sources (on any platform accessible to SAS, in any format accessible to SAS)
- Targets (on any platform accessible to SAS, in any format accessible to SAS)
- Transformations that specify how data is extracted, transformed, and loaded from a source to a target
- Jobs that organize a set of sources, targets, and transformations
- Source code generated by SAS ETL Studio
- User-written source code.

The 'single point of control' notion is a large selling point. Whereas an experienced and well-versed SAS programmer could duplicate a lot of the functionality and capabilities provided by SAS ETL Studio, the end result is typically idiosyncratic and difficult to share among a large number of users. And the graphical interface helps make the process more understandable to end-users in a much shorter period of time, thus helping to alleviate, but not eliminate, the problem a corporation has replacing the knowledge in someone's head when that person leaves the company.

SAS and SAS ETL Studio provide a number of features that support enterprise data management, which include the following:

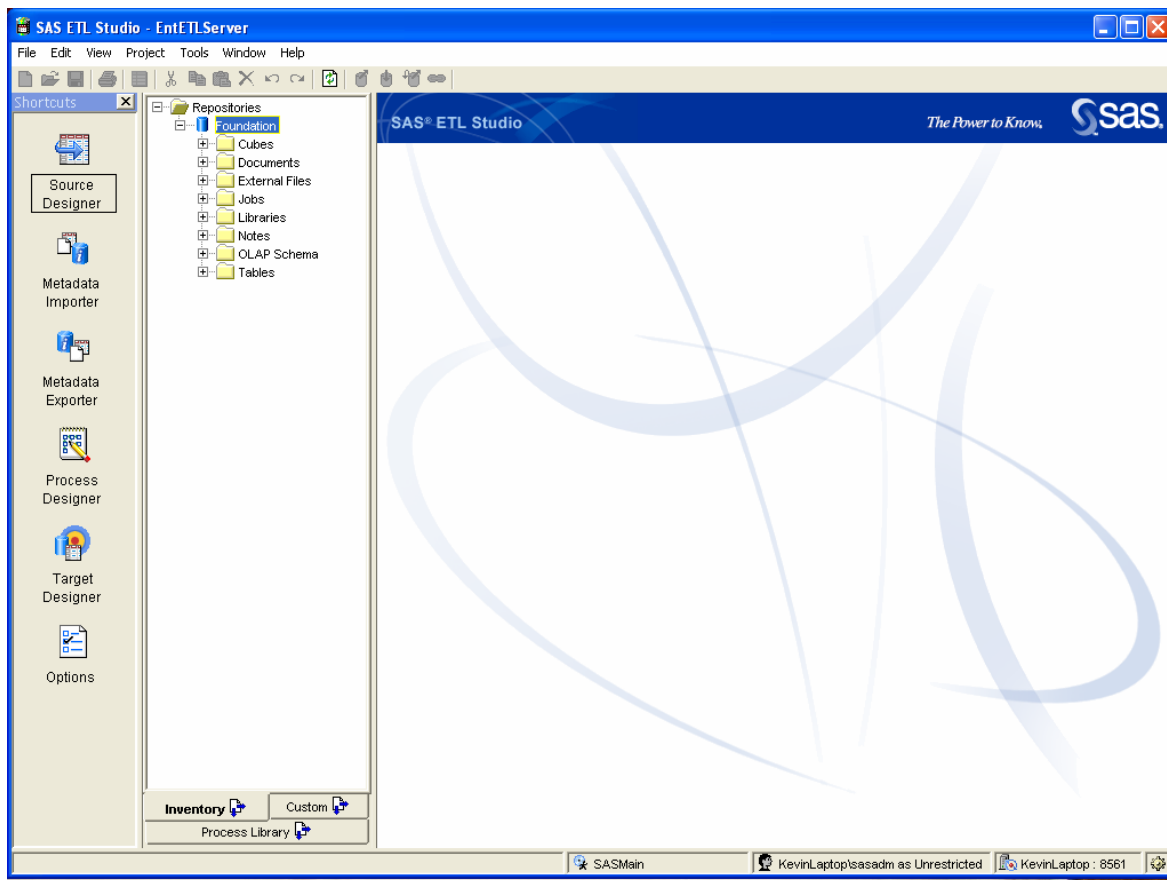
- a metadata architecture that enables applications to share metadata for servers, libraries, tables, and other resources.

- import and export of metadata in Common Warehouse Metamodel (CWM) format, with optional bridges being available for other formats.
- source control for metadata.
- metadata backup.
- access control for users and groups.
- support for multiple work environments—separate environments for development, testing, and production, for example.
- multi-tier (N-tier) support for processes that flow across multiple servers.
- data quality processes for SAS ETL Studio jobs.
- scheduling for SAS ETL Studio jobs.
- status handling for SAS ETL Studio jobs. The status of a SAS ETL Studio job or a transformation within a job can be automatically e-mailed, written to a file, or sent to an event broker that will pass the status code to another application. You can also use status code handling to capture job statistics, such as the number of records before and after the append of the last table loaded in a job.
- export and import of SAS ETL Studio jobs.
- support for slowly changing dimensions, which enables you to track changes to data that occur over time.
- support for stored processes. You can save a SAS ETL Studio job to a file for the SAS Stored Process Server to execute later.
- a Cube Designer wizard for creating OLAP data stores.
- optional data surveyor wizards that enable you to build SAS ETL Studio jobs that read data from SAP, Siebel, PeopleSoft, Oracle, and other enterprise application vendors.

What do you need to run SAS ETL Studio? Software wise, at a minimum you need to have a SAS Metadata Server and a SAS Workspace Server up and running (note that both these servers come with SAS/BASE) and SAS ETL Studio of course. In terms of personnel roles, you will likely want to have a metadata administrator (the person who administers SAS Management Console) in addition to the ETL Studio developers (as many as you need and license). Note that ETL Studio has built in check-in/check-out control that allows you to make sure that the same component is only being altered by one developer at a time.

Quick Overview of the Main Windows in SAS ETL Studio

SAS ETL Studio is a java application that has a very intuitive look and feel. For experienced MS Windows users, the toolbars and menus will be a snap to locate and use. Here is a pretty bare bones picture of what the default screen looks like:



At the far left of the screen is a shortcut column. Personally, I do not find it useful and close it by default. The next section contains the Inventory, Custom, and Process Library tree views (there is also a Metadata view that I have turned off and thus it does not display here). These 3 trees will prove invaluable to you. In a nutshell, here is what they provide:

Inventory tree – Think of this view as the one that SAS Maintains for you. Anytime you add an object to your metadata repository, SAS will store it in the appropriate folder. Add a new Source or Target table, and they will show up under the 'Tables' folder. If someone wants to review what is in your repository and you've not had time to explain your 'Custom' view to them, this is the place they will likely start.

Process Library tree – This view contains a list of the transformations that you may utilize in your ETL jobs. It contains both the set of folders that contain the transformations shipped with the software in addition to any custom transformation folders that you create.

Custom tree – The custom view is just that, custom. By default, items get put into an 'Ungrouped' folder as objects are added to the repository. You may create a folder schema that makes sense to you. As I've grown

more familiar with the software, I find myself gravitating towards using this window more and more.

Process Designer Window – The large panel on the right side of the screen is where you will design and create your jobs which will resemble flowcharts. SAS ETL Studio organizes sources, targets, and transformations into jobs. SAS ETL Studio uses each job to generate or retrieve SAS code that reads sources and creates targets in physical storage.

Creating a Job in SAS ETL Studio

For purposes of demonstration, let's create a very simple example. The objective of the job we want to create is to append a couple of source data sets that contain furniture sales information (sashelp.prdsal2 and sashelp.prdsal3), create a custom transformation that puts all character variables in upper case, and then store the aggregate file in our target table.

1) Register the source and target libraries.

Because SASHELP is not a source library by default, let's copy the PRDSAL2 and PRDSAL3 tables to an arbitrarily determined library, "c:\temp\mysources" with a libref of PROD_SRC.

The creation and registration of the library can be accomplished in Management Console or in ETL Studio. Within ETL Studio, go to the Inventory tree and right click on 'Libraries'. From the popup menu, select 'New' to start the New Library Wizard. On the first screen, select 'SAS Base Engine Library'. On the second screen, provide a descriptive name for the library (up to 60 characters and spaces are OK). On the next screen you will provide the actual LIBREF (or LIBNAME), verify the engine version, and provide the path relative to the server being utilized. On the next screen, you can select the servers that you want to make the LIBREF available to, assuming you have more than one server involved. The last screen is a summary screen which summarizes your choices and processes them when you select 'Finish'.

Repeat the above steps to create a target library named PRODTARG that points to "c:\temp\mytargets".

2) Register the Source Tables

The library that we registered in step one can have 1 or 100 tables within it but we cannot utilize the tables as source tables in our jobs until we have registered the tables individually. Using the Source Designer Wizard, we will register the two aforementioned tables. From the toolbar, select Tools-Source Designer. The first screen of the wizard will ask you for the data type, in this case we will use SAS.

The second screen of the Source Design Wizard asks you to indicate the library where the table resides. We will use the library created in step one. Note that if a message is returned indicating no tables were found, you may need to restart your workspace server so the libref can be generated. In our case, the PRDSAL2 and PRDSAL3 tables should be found and by choosing 'Select All' and then 'Next' we can import the metadata regarding both tables at once. On the next screen, select the default 'Foundation' repository as the location for the metadata. If you accept all the info present on the summary screen, select 'Finish' to create the metadata.

After a very brief amount of processing, the screen will return to looking much like the default when you first opened ETL Studio. In order to verify that your tables were registered, go to the Inventory tree and expand the Tables folder. You should now see the PRDSAL2 and PRDSAL3 tables present. You can view the data and inspect their properties to see what type of information is being stored about the tables. Note that your table has not been copied, the physical table still exists in its original location. Also notice that in the Custom Tree, the two tables now appear under the 'Ungrouped' folder.

3) Create the target table metadata

Although you have not yet created a physical copy of what will be your target table, you can do so now if you have a clear plan of what variables you will be adding or dropping. You can also perform this step later in the process. In our case we will simply be dropping a few date variables (MONTH MONYR, and DATE).

Select Tools-Target Designer from the menu bar. On screen one of the Target Designer wizard, indicate that you want to create a Target Table. On screen two, provide a label for your table (up to 60 characters) and an optional description (up to 200 characters).

Screen three asks you to select the columns from each table that you would like retained in the target table. Screen four allows you to import additional columns from other sources, create computed columns, as well as created indexes.

Screen four asks you to specify the library and valid SAS name of the target table that will be created. In addition, you can specify options such as compression, encryption, and other data set options.

Screen five asks you to specify the repository in which to store the metadata.

Screen six is the summary screen. Upon hitting 'Finish' the metadata will be created.

Again, you will need to look under the Tables folder in the Inventory tree to see that your target table metadata has been created.

4) Start the New Job Wizard to create the job shell

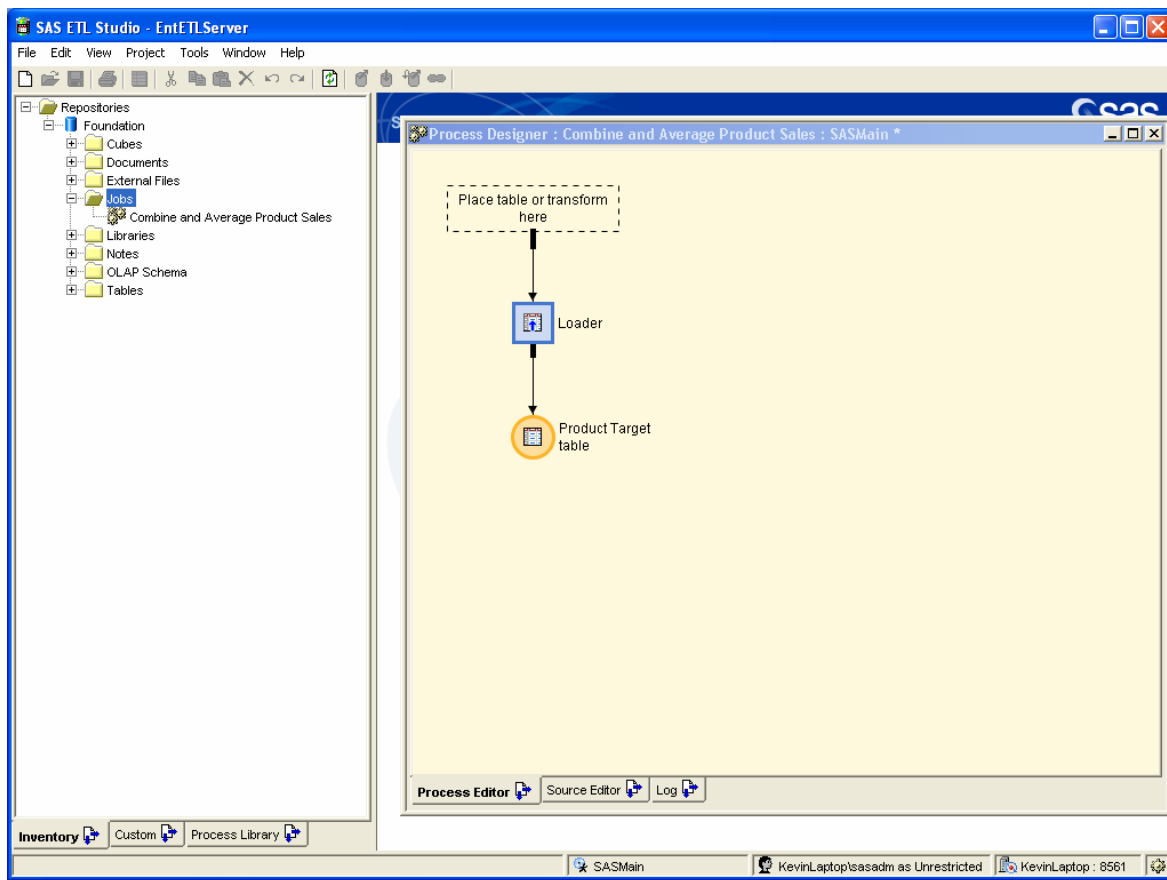
You can begin creating a new ETL job in one of several ways. Two of the most common are: a) Select 'Tools – Process Designer' from the menu bar, or b) Right click on the Jobs folder in the Inventory tree and select 'New'. Either method kicks off the 'New Job Wizard'.

Screen one of the wizard simply asks you for a logical job name. Make this name something very easily recognized. In time, you may want to develop a nomenclature for naming your jobs so that the organization is readily apparent as the number of jobs grows.

The second screen asks for the target tables to be loaded in the job. We will specify the metadata object created in step 3 above.

The third screen asks for the repository that you want to store the job, in this case it will be 'Foundation'.

The fourth screen is the familiar summary screen and hitting 'Finish' brings up the Process Designer window which has the beginning of our job flow.



Note that the termination point of our process flow is the target table. We can build as much as we want/need above that point to build the job that we need.

5) Add the two source tables to the job

In the Process Library tree, drag the 'Append' object from the Data Transform folder onto the node that says 'Place table or transform here'. This action will add an append object to your flow and above that will be two 'Place table or transform here' boxes (you can place more than two without problem if needed). For our purposes, we will drag our two source tables (PRDSAL2 and PRDSAL3) onto these two boxes.

6) Create a custom transformation to put all character variables in upper case

The Transformation Generator wizard enables you to create user-written, SAS code transformation templates and make them available in the Process Library tree. The concept is somewhat similar to a SASAUTO library in that you can

create reusable code but the interface and ease of use in ETL Studio is a big plus for developers.

Select 'Tools-Transformation Generator' to start the wizard. Screen one asks for a logical name, we will use 'Uppcase Char Variables', for the transformation and asks where you would like to store it. It will help with organization if you create something like a 'Custom Transformations' folder and you will likely create many subfolders within that folder over time.

Screen two asks for the SAS code that will be generated by the object. In this case, we insert:

```
data &_output;  
  set &syslast;  
  array cvarup(*) _character_;  
  do i=1 to dim(cvarup);  
    cvarup(i)=upcase(cvarup(i));  
  end;  
  drop i;  
run;
```

Note that the &_output and &syslast macro variables are handled internally during a job flow (you do have some control over their use). SYSLAST simply represents the last data set created by the previous node in the job and _OUTPUT resolves to some obscure SAS name that is unlikely to conflict with a name you would make up yourself. Although an individual transformation may contain many data steps, it is good practice to have the last data set created use the _OUTPUT macro naming convention.

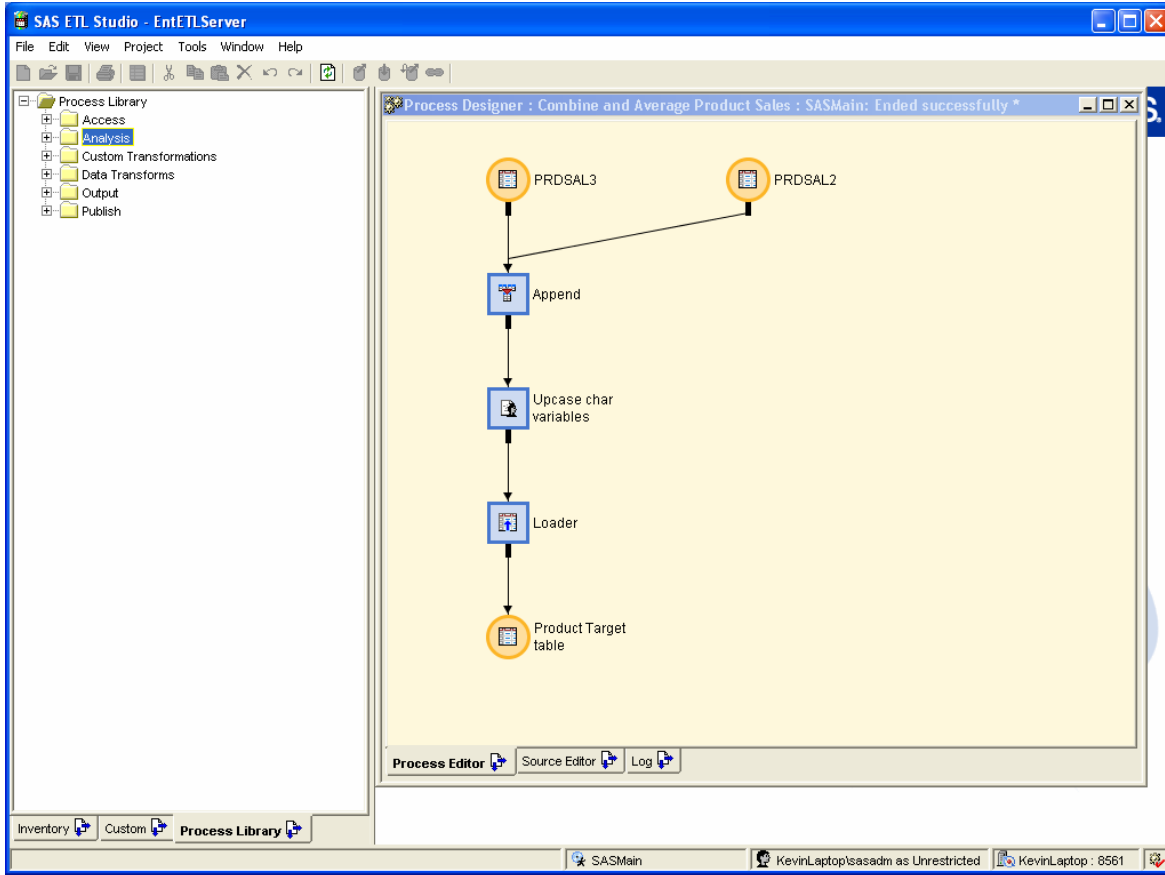
Screen three asks for any input parameters that you want to pass to the code. These parameters will need to be set each time you use the object in a job flow.

Screen four asks how many input and outputs the transformation will support. Most transformations will have one input and one output however this option provides a lot of flexibility in terms of splitting a flow or creating a transformation that requires no input.

Screen five is the summary screen and hitting 'Finish' creates the transformation.

At this point, we will insert our new transformation into the job flow by going to the Process Library tree and locating our new transformation. Once found, left drag the transformation over the arrow under the 'Append' object and release.

Here is the completed version of our simple job:



During the presentation, we will review the properties of the job including options for running the job, scheduling the job, how to pre- and post-process custom code, and saving jobs to run as stored processes.

Summary

Hopefully this short introduction shows you the relative ease that you can get started with ETL Studio and make it an important part of your organization. The control and standardization of routines that it provides as well as the self-documentation can make it a wise investment and make it easier to share code among a group of users.

My initial experiences have been quite positive although there are many improvements to the software that I would like to see implemented. The software is still in its early stages and the incremental improvements made thus far in the early releases have been substantial so I don't doubt that the software will improve dramatically.

Here is a short bullet list of items I would like to see implemented:

- True version control
- Better support for documenting and reporting over time how a job process was altered.
- Better note handling for documenting decisions. Currently notes are retained at the instance level whereas frequently I find that I would like a note to be attached to all instances of a custom transformation.
- More user-defined metadata options for an individual table at the column level.
- Allow process flows to be copied to Visio or MS Word.
- Make the options screen on the transformation generator more flexible so that many parameters could be set for any given option line and allow the options to be data driven (e.g. restricted to certain ranges, 'Yes-No', etc.)
- Allow rearranging of transformation nodes within a given Process flow
- Allow for parameters to be set at the job level without having to use Preprocessing code file option.

Author

Kevin Davidson, Ph.D.
FSD Data Services, Inc.
2020 Southwest Freeway, #206
Houston, TX 77098
713-942-8436
kevind@fsd.nu