Paper 210-2007

The Ins and Outs of SAS® Data Integration Studio

Chris Olinger, d-Wise Technologies, Inc., Raleigh, NC
Tim Weeks, SAS Institute, Inc., Cary, NC

ABSTRACT

The SAS® Version 9 suite of tools provides many new resources for managing and manipulating data. SAS® Data Integration Studio (DI Studio) is one of the many SAS® Version 9 applications in the Data Transformation and ETL space. DI Studio is a new breed of SAS® tool that lets the user visually define executable processes, which up until now had to be created and deployed using traditional SAS programming and operating system techniques. This tutorial explores the different aspects (both good and bad) of the ETL tool using a simple case study, and is intended for beginner to intermediate SAS programmers.

INTRODUCTION

Anyone who has used SAS to process and transform data has invariably said to themselves "there has to be a better way." SAS is a great data processing language, but it is source code intensive. As such, you need a good set of SAS programmers to implement and support a data warehouse. Code must be designed, developed, tested, and then maintained. Each of these steps requires a certain amount of effort and support (debugging SAS code comes to mind). Depending on the complexity of the task, this can be a daunting undertaking.

DI Studio is a tool specifically designed to help simplify the ETL process. The idea behind the tool is that the various steps in an ETL process can be packaged into predefined units. These units of execution can be strung together, in a visual manner, to create a job. Jobs are then scheduled and automated. Each code unit can either use DI studio generated code, or code provided by the user. The more that standardized code units are used, the more standard your jobs become. Obviously, if the SAS generated code is up to snuff, then the time to market for your project is decreased.

Along these lines, source data and outputs can live on any number of systems. DI Studio supports all of the major databases, as well as text inputs and a myriad of ODBC sources. Connecting to these systems is relatively painless once the appropriate setup has been performed in the Metadata Repository. As well, schemas and models can be designed in external tools like ERwin and imported for use.

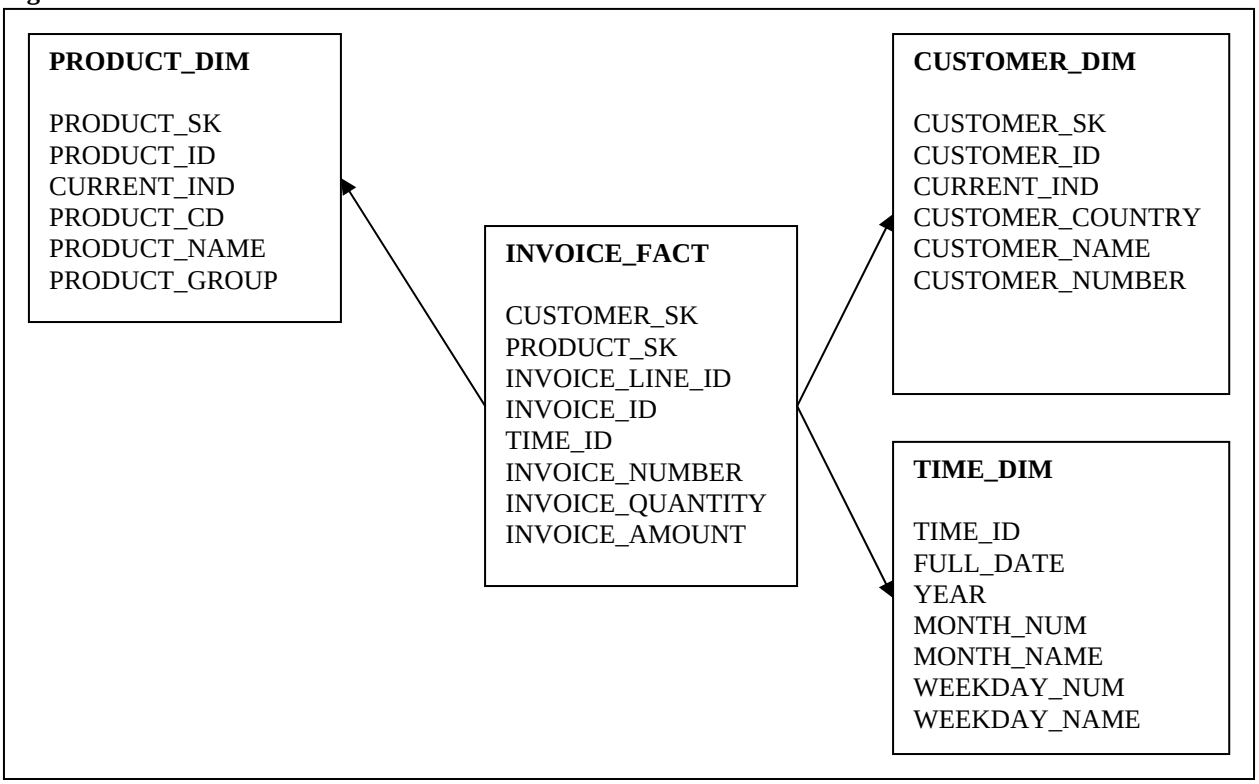
Each process is managed and created according to metadata. At first, this may cause some confusion – you are not working directly with the tables, columns, and jobs. You are working with the metadata that defines the tables, columns, and jobs. You do not actually generate the code that makes up the job until it is tested. Given this, DI Studio can be thought of as a metadata manipulator. Once you have mastered this notion, you will realize that jobs can actually be created without any source data at all, allowing the user to design a warehouse before your source systems are ready!

So how does the tool stand up? In the end, the real test of any tool is how well is does what it claims to do, and how well it helps the developer do his or her job. If it is more convenient to write the code by hand then what is the point? Over the last two years, we have been working with DI Studio to retrofit an existing enterprise warehouse. We hope to answer some of these questions with this paper.

THE EXAMPLE

Our example is a simple data warehouse. The ETL process we will create will transform source sales data from a sporting goods store. The outputs of the ETL process will be data that resides in a star schema, with a fact table and three dimensions. We will use source data that exists in SAS tables, and the outputs from the process will be SAS tables. Visually, our warehouse will looks like the following:

Figure 1



Source tables consist of the following:

- COUNTRY – Country codes to country names mapping
- CUSTOMER – Customer data including name and country
- INVOICE_LINES – Invoice line level data
- INVOICE – Customer invoice data and invoice dates
- PRODUCT_GROUP – Product groupings and designations
- PRODUCT_GROUP_X_PROD – Product groupings to product ID designations
- PRODUCT_LIST – Product listing table containing product names
- DENORMED_SOURCE – A de-normalized table containing a de-normalized version of all of these tables

Our mission is to turn these source tables into a functioning data warehouse using DI Studio. Before we can do this, however, we need to explore the required steps to setup the metadata required for DI Studio to function.

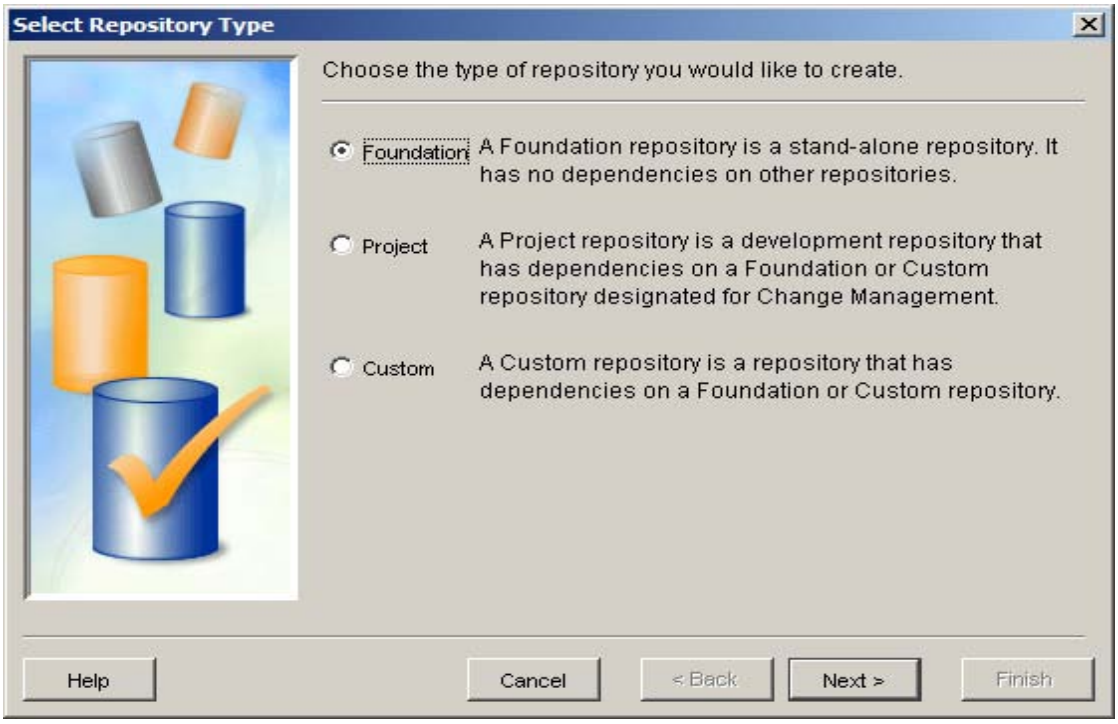
SETUP AND THE IMPORTANCE OF METADATA

DI Studio runs off of metadata. This metadata includes input libraries, source table information, and target table information. In order to use this metadata we must first set up the system correctly. To do this, we must run the SAS Metadata Console, or SMC. These steps are usually handled by a data manager or support staff. However, in a development environment, you often need to make configuration changes (for instance, adding a new libname to the system). In this case, you can either wait for your support staff to do it for you, or you modify the metadata yourself.

Note, these SMC modifications are predicated on correct setup of metadata users. This topic is fairly involved so we will skip it here in this paper. Just remember this, if things are not working right, more than likely you have a permissions problem with a given user that will most likely need to be fixed before continuing.

We will start with an empty metadata repository. A repository is a named collection of metadata, nothing more. Repositories can be of type Foundation, Project, or Custom. A Foundation repository is a master repository. A Project repository is a user based project repository. It used so that individuals can edit and test in a private environment. The

Project repository “looks through” to the Foundation repository. Once created, a Project repository gives the user the ability to check in and check out objects. From the Metadata Manager node in the SMC, select the active server, and then select Add Repository. The following screen will appear:



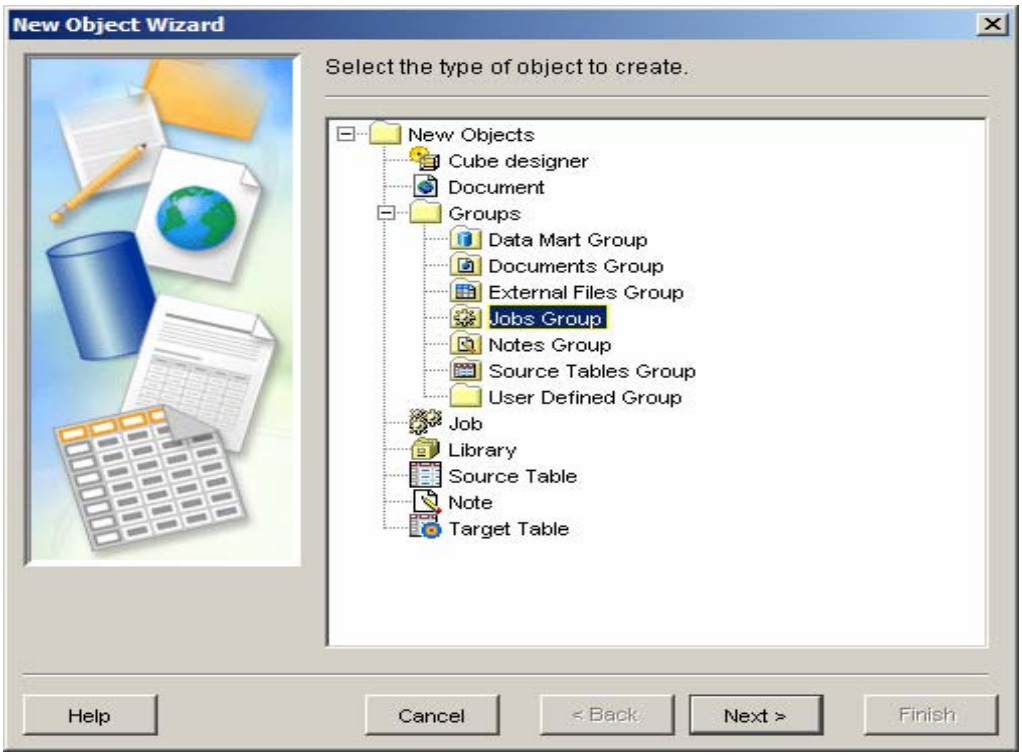
We will need a Foundation repository (if not already setup) and a Project repository. Once these are created, we can use DI Studio to create some rudimentary structure.

By default, a repository is empty. You must add any structure that will help with the organization of your metadata. For our needs we will need a folder for:

- Libnames
- Source Data
- Target Data
- Jobs

In DI Studio, select the Project tab, select the Project repository name, then select New Object.... You should see a screen that looks like the one below. Create a Jobs folder, a Source data folder, a Target data folder, and a folder for libnames. After you have created the folders, select “Check In The Repository”. This will commit the structure to the Foundation repository.

After we have created our folders, we can create our necessary libnames.



LIBNAMES

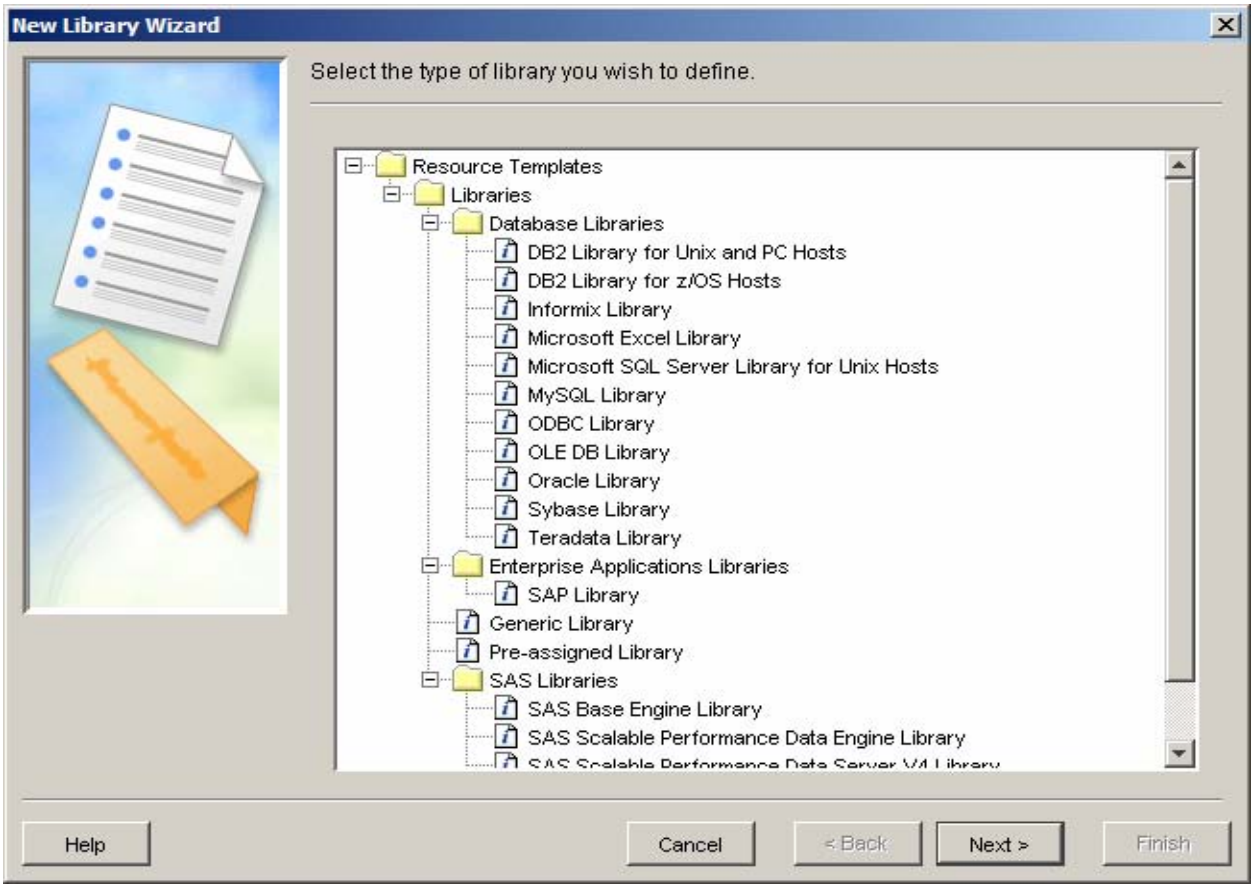
Standard SAS programming techniques have dictated for many years that library references (libnames) should be global data. If you code to libnames then your SAS program becomes more portable than if you reference a file directly by name. For Version 9, SAS gives you the ability to declare and manage your libraries as part of the metadata repository. This centralized location gives data managers the ability to manage data references outside of a standard SAS program, as well as share libnames across SAS products. DI Studio leverages these shared libraries heavily. You can still generate your own libname statements, but DI Studio will not be able to reference metadata for these libraries – you must create the libraries using the SMC first.

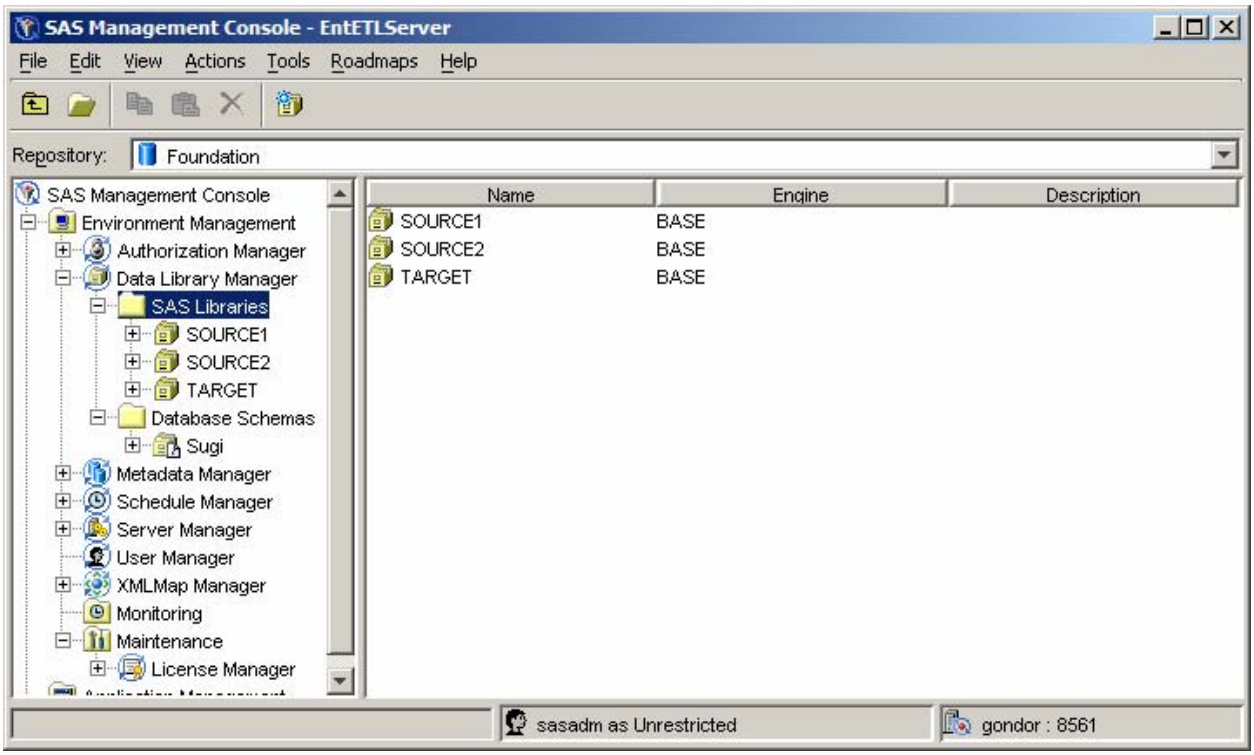
Given this, the coding model must change. You need to contact your data manager, or support staff, to add and change library references. This can sometimes be a nuisance, but in the end is worth it. Note, for development efforts, we recommend that individuals be given access to the SMC to add and change libraries on their own. It is much easier to modify a library yourself than to wait on another person.

To add a library to the SMC, select the Data Library Manager node and then select SAS Libraries. From there you can add a new SAS library. As you can see from the figure below, there is quite an array of storage types that you can map a library to. Select SAS Base Engine Library. This will let us create a libname that points to standard SAS data sets. For our example we are going to create three libraries: two for source data and one for target data. At the completion of this step the SMC should look something like the picture on the next page.

When we have completed our library setup we can close SMC and move back to DI Studio.

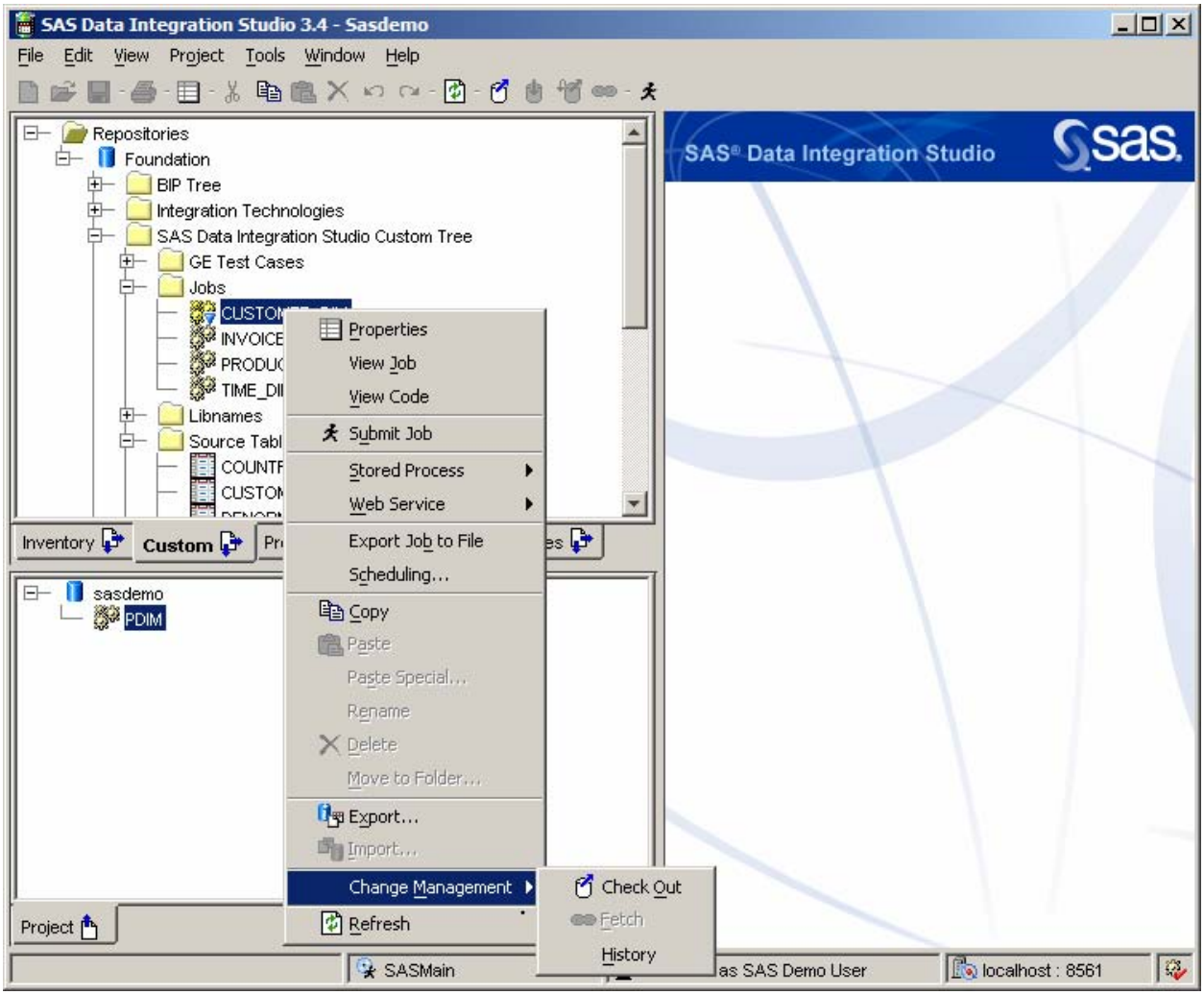
Note, you do not have to write code in your DI Studio jobs to generate a libname that is managed by the SMC. Any table that is used in the Job automatically references the libname that it is associated with. When source code is generated the libname is automatically added to the code.


New Library Wizard
Select the type of library you wish to define.
Resource Templates
Libraries
Database Libraries
DB2 Library for Unix and PC Hosts
DB2 Library for z/OS Hosts
Informix Library
Microsoft Excel Library
Microsoft SQL Server Library for Unix Hosts
MySQL Library
ODBC Library
OLE DB Library
Oracle Library
Sybase Library
Teradata Library
Enterprise Applications Libraries
SAP Library
Generic Library
Pre-assigned Library
SAS Libraries
SAS Base Engine Library
SAS Scalable Performance Data Engine Library
Help
Cancel
< Back
Next >
Finish


SAS Management Console - EntETLServer
File Edit View Actions Tools Roadmaps Help
Repository: Foundation
SAS Management Console
Environment Management
Authorization Manager
Data Library Manager
SAS Libraries
SOURCE1
SOURCE2
TARGET
Database Schemas
Sugi
Metadata Manager
Schedule Manager
Server Manager
User Manager
XMLMap Manager
Monitoring
Maintenance
License Manager
Name
Engine
Description
SOURCE1 BASE
SOURCE2 BASE
TARGET BASE
sasadm as Unrestricted
gondor : 8561

PROJECTS

The project repository in DI Studio is the developer's area for making changes to existing metadata objects and for creating new metadata objects. This repository is separate from the Foundation repository so that changes to the Project repository have no impact on the rest of the metadata. To get objects into the Project repository a developer either checks out existing objects from the Foundation/Custom repository, creates new metadata objects via the "New Object Wizard," or fetches the object.



As soon as checked out metadata objects are "pulled" to the project repository, a lock is put on the object in the foundation repository to ensure others cannot check out the same object. The developer can then make changes without fear of causing problems for anyone else. To get objects back into the foundation repository there are a number of options. First, the objects can be checked in, which releases the locks and saves the changes to the foundation. Second, an undo check-out can be performed, which releases the locks and discards any changes. Finally, the object can be destroyed; this permanently deletes the object from the foundation repository.

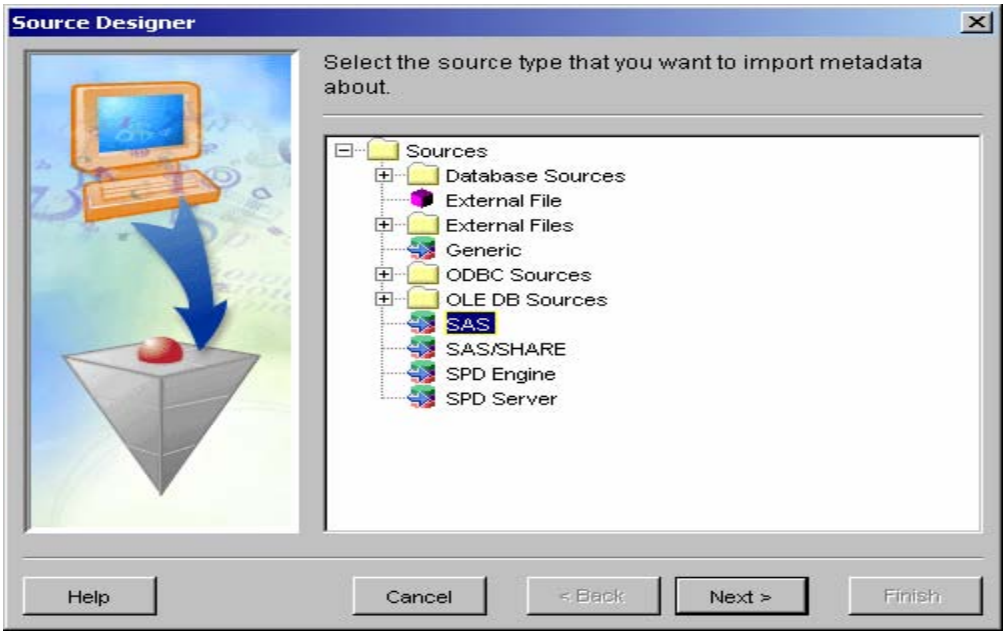
NOTE: All objects in the project repository have to be checked in at once! This is a limitation that is particularly frustrating as you often have many different changes going on at once. Often, you would like to check in just one job or entity. Make sure you save often, and keep only a small amount of work in your repository.

Fetching an object allows the developer to get objects into the project repository in read only mode. This is useful when objects are needed but the developer knows no changes are required. An added benefit is that no locks are placed on the object in the foundation repository. This allows other developers access to it. To remove the object from the project repository it must be deleted. This action only deletes the object from the project repository not the foundation.

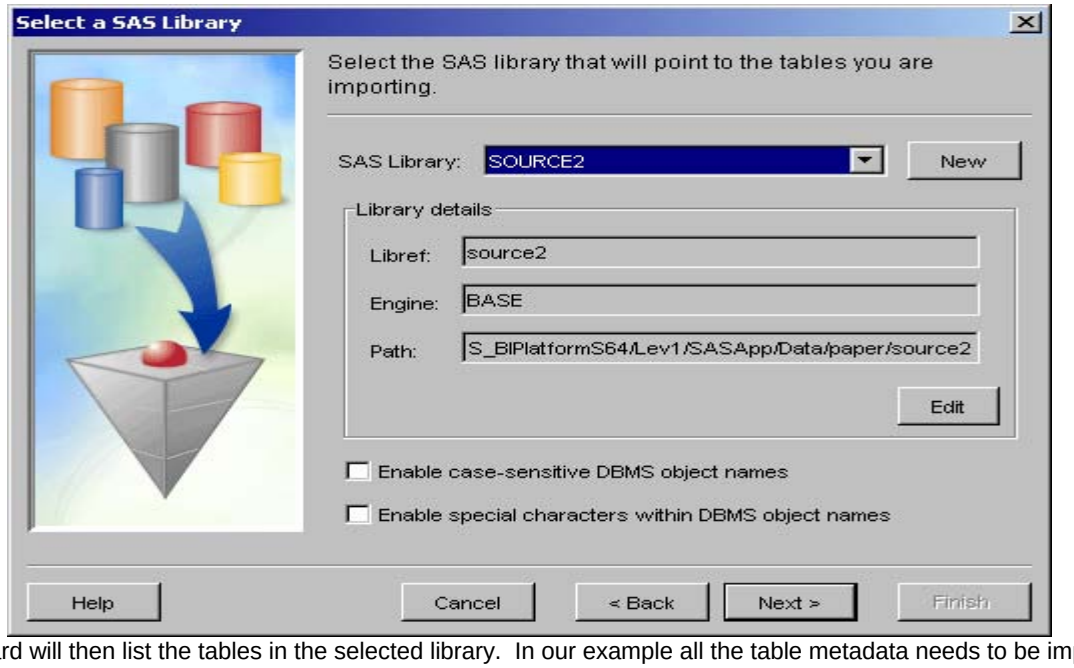
CREATING TARGETS AND SOURCES

SOURCES

A source in DI Studio can be a SAS dataset, DBMS table, external file, or any other data structure that SAS can access. Metadata about the source data must be created in DI Studio (or the SMC) before it can be used in jobs. To simplify the process there are a number of wizards for the different data types available. Metadata about the libname for the source must be entered before using the source designer wizards. To start the designer wizard, select Tools -> Source Designer from the DI Studio desktop. The wizard will then provide a list of the available source types to choose from. In this scenario we will be pulling metadata from a SAS source, so the SAS icon is selected.

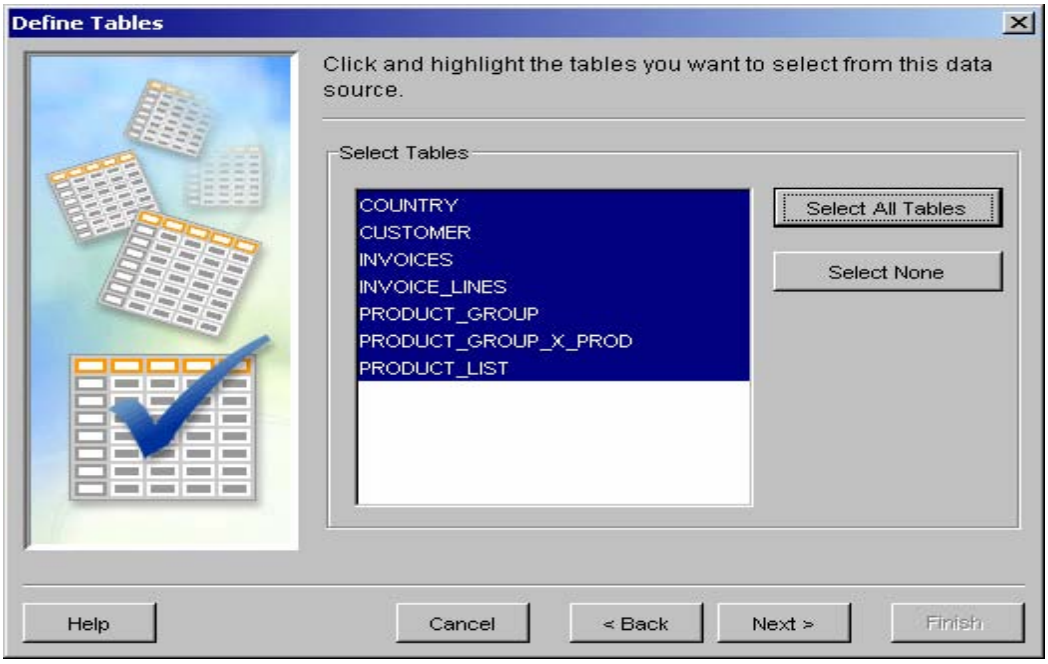


The next step is to select the library where the source tables reside. The wizard will list all the registered libraries in the metadata of the type selected.

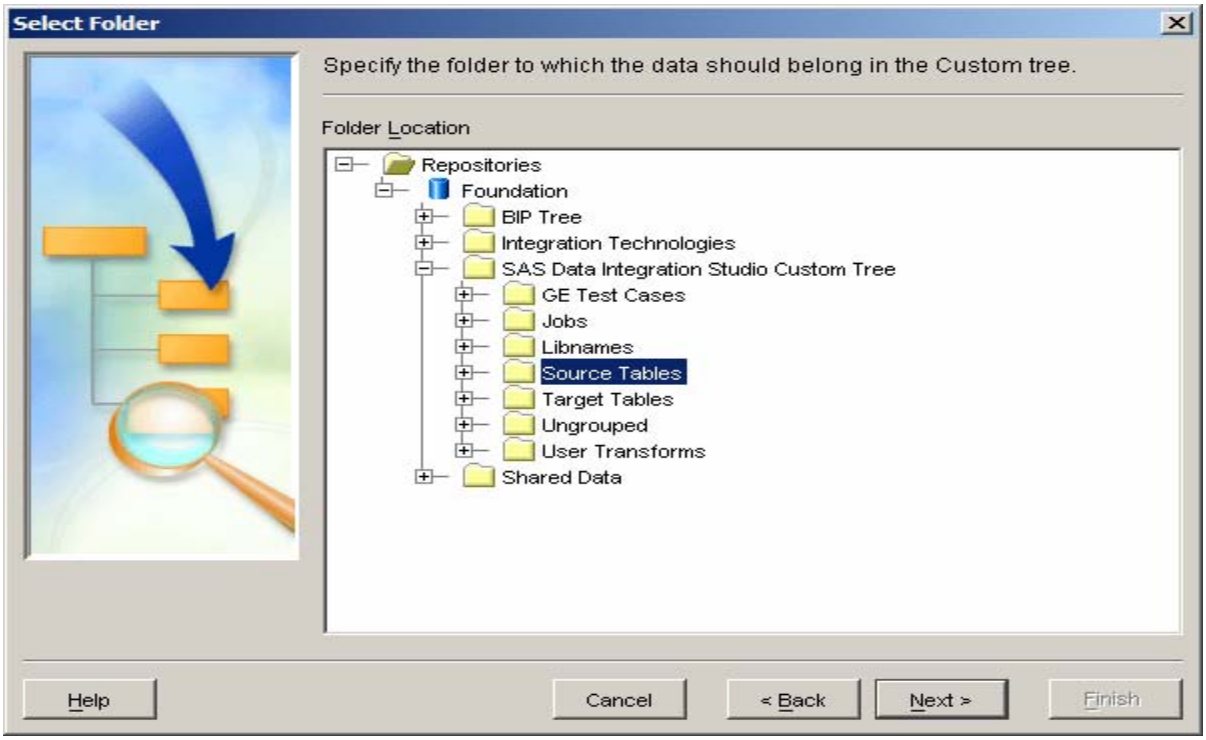


The wizard will then list the tables in the selected library. In our example all the table metadata needs to be imported

so clicking the “Select All Tables” will highlight all the tables.

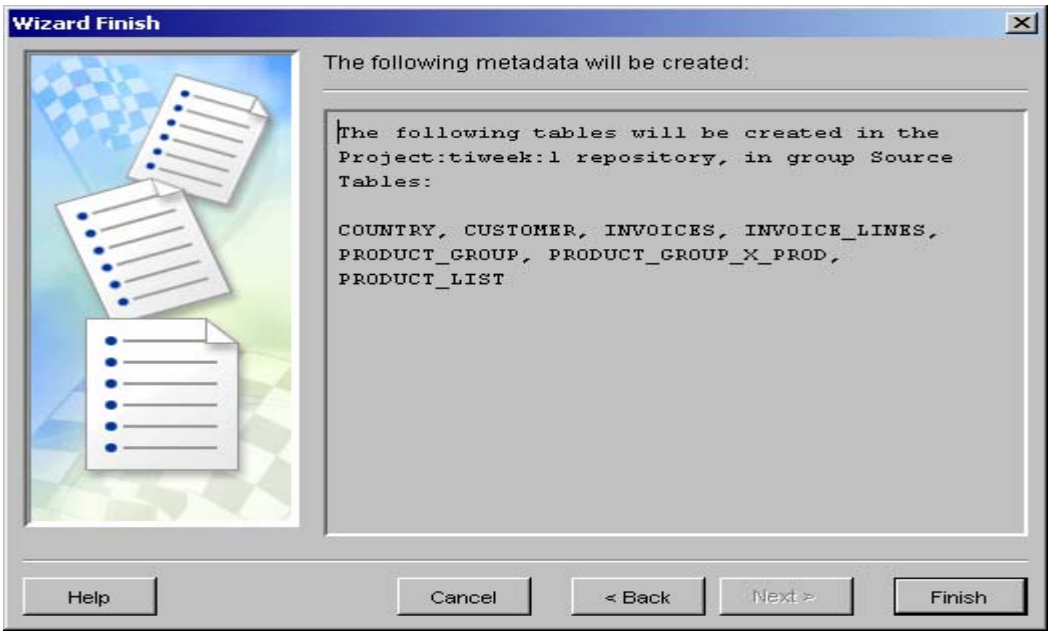


Now that the tables have been selected the next step is to choose where the metadata should be stored. In this example we have created a “DI Studio Paper” folder then a “Source Tables” sub-folder where the table metadata will be stored. Folders have to be created and checked into the foundation repository before beginning the source designer.



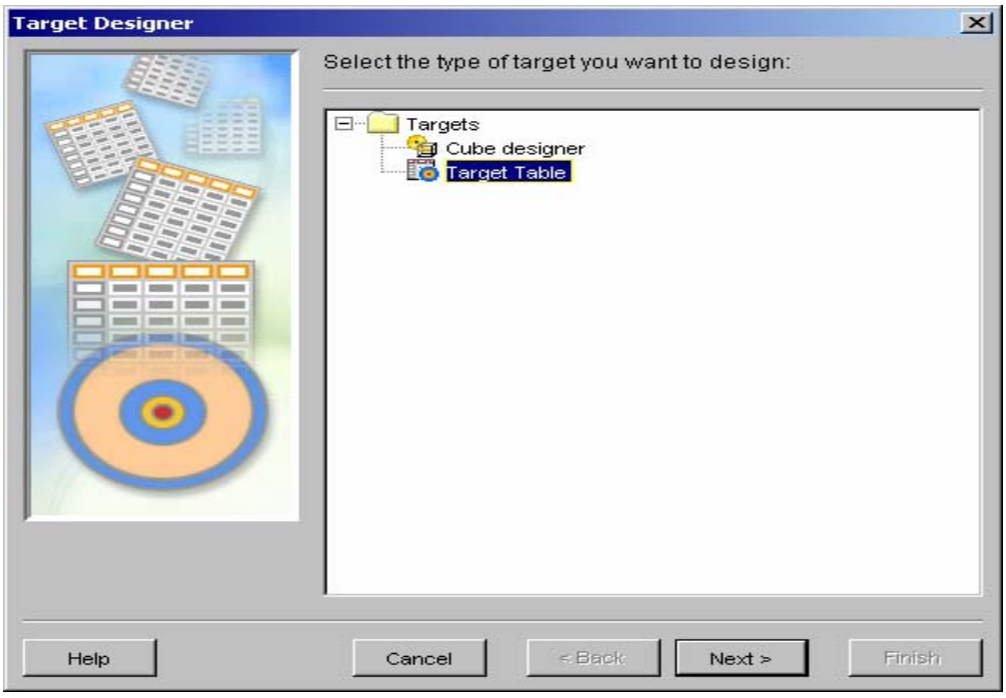
The final step in the process is the summary page which displays a list of the selected tables, the folder where the

metadata will be stored, and the repository where the metadata will be created.

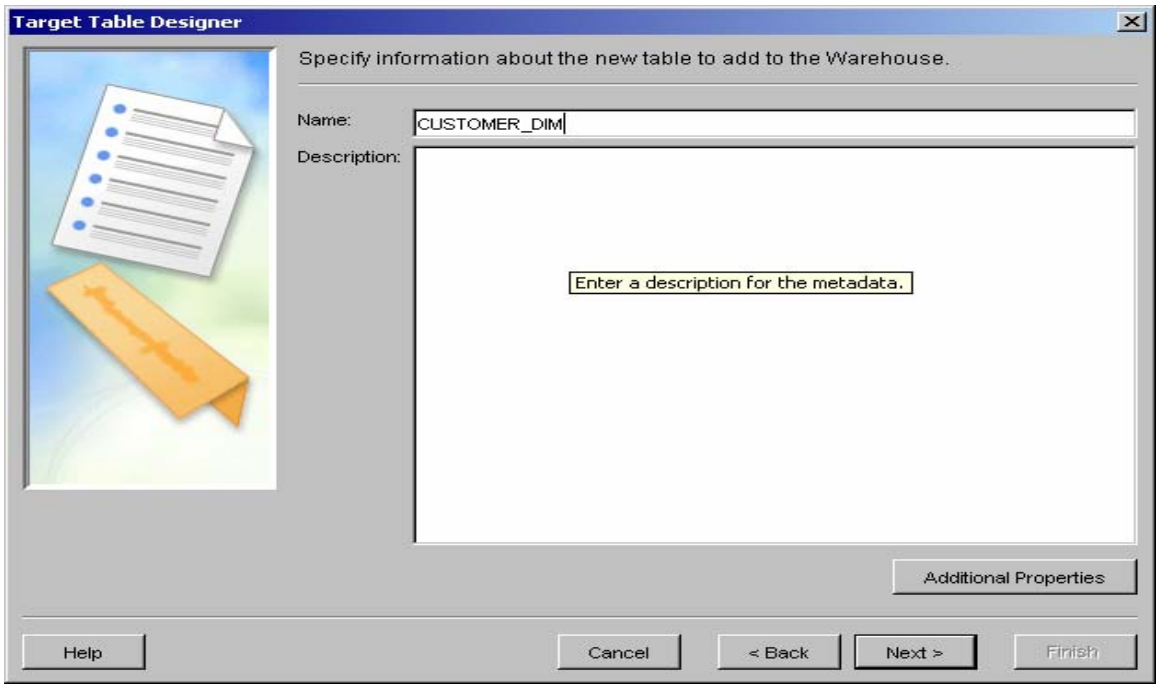


TARGETS

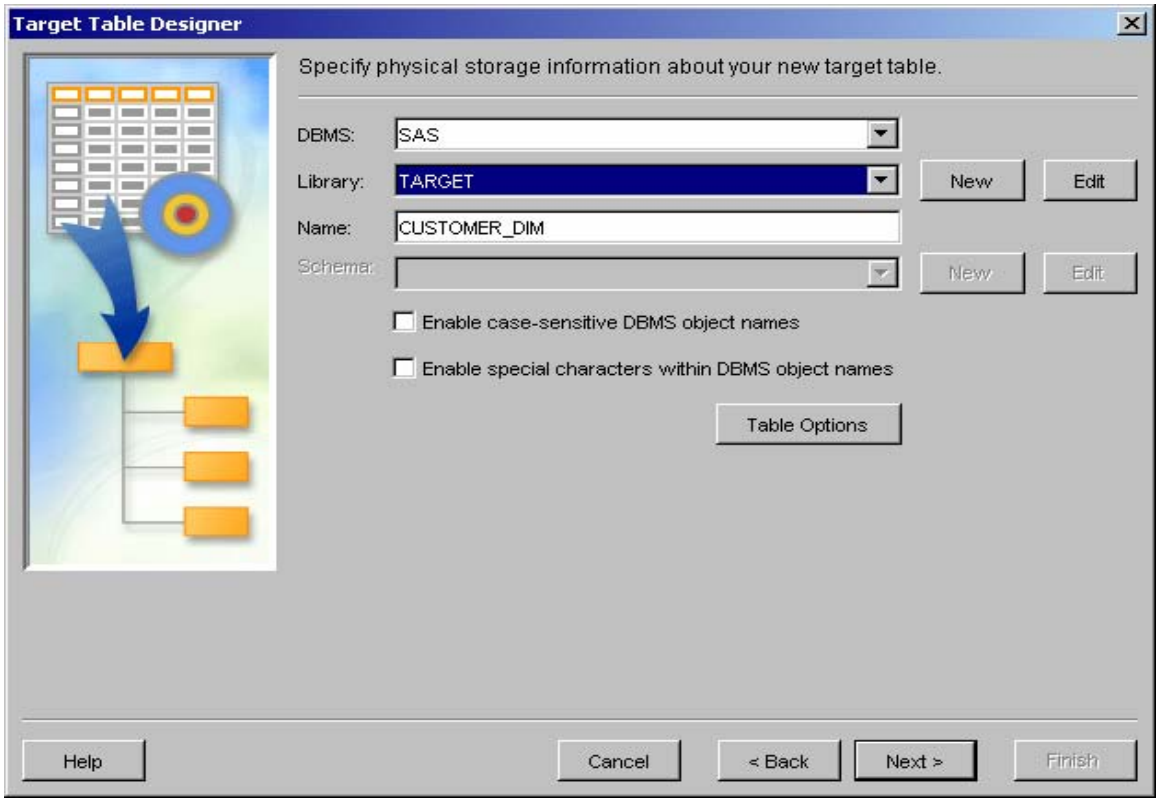
A target in DI Studio is very similar to a source in that it is metadata about a SAS dataset or DBMS table. The difference is that a target is the metadata about the structure and location of the table the jobs will be creating and loading. These objects may not exist yet so the "Target Designer" is a little different than the "Source Designer". Keep in mind that if the target structures already exist nothing prohibits you from using the "Source Designer" instead. This is sometimes a faster way to pull in the necessary metadata. In the example that follows we are assuming the target tables do not yet exist. To start the designer wizard, select Tools -> Target Designer from the DI Studio desktop. The first step in the process is to select which type of target you need to create. There are only two options: Cube or Table. The target in this example is a SAS dataset so the Table option is selected.



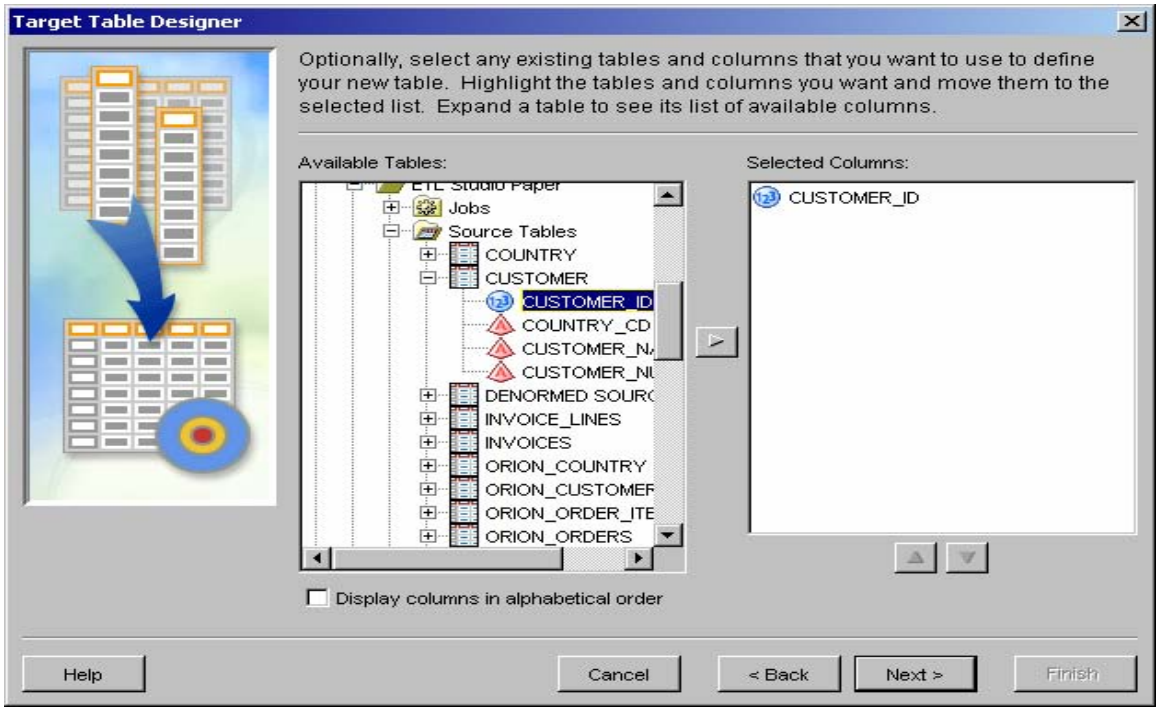
The next step is to provide the name of the target table being created and a description.



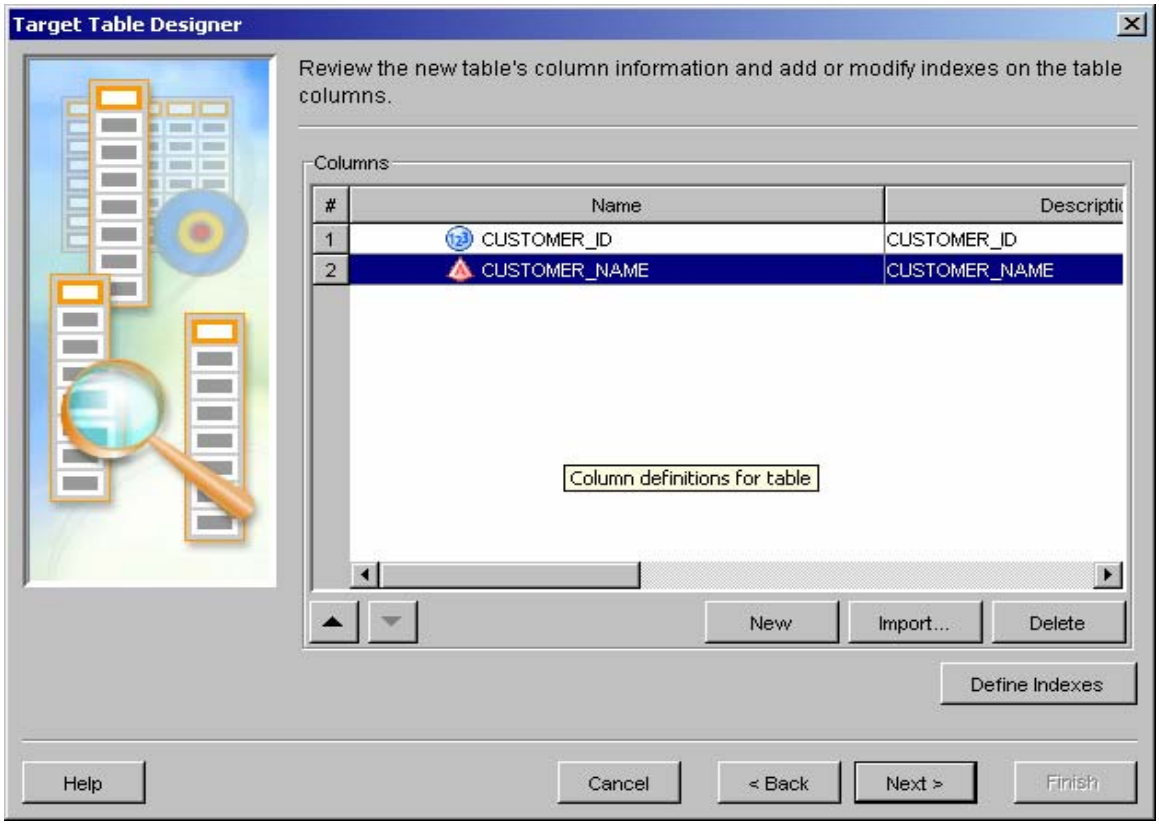
You then need to select the library where the table will be stored.



The fourth step gives you the opportunity to import column definitions from existing table metadata. The DI Studio groups are displayed to help organize the metadata to simplify access.

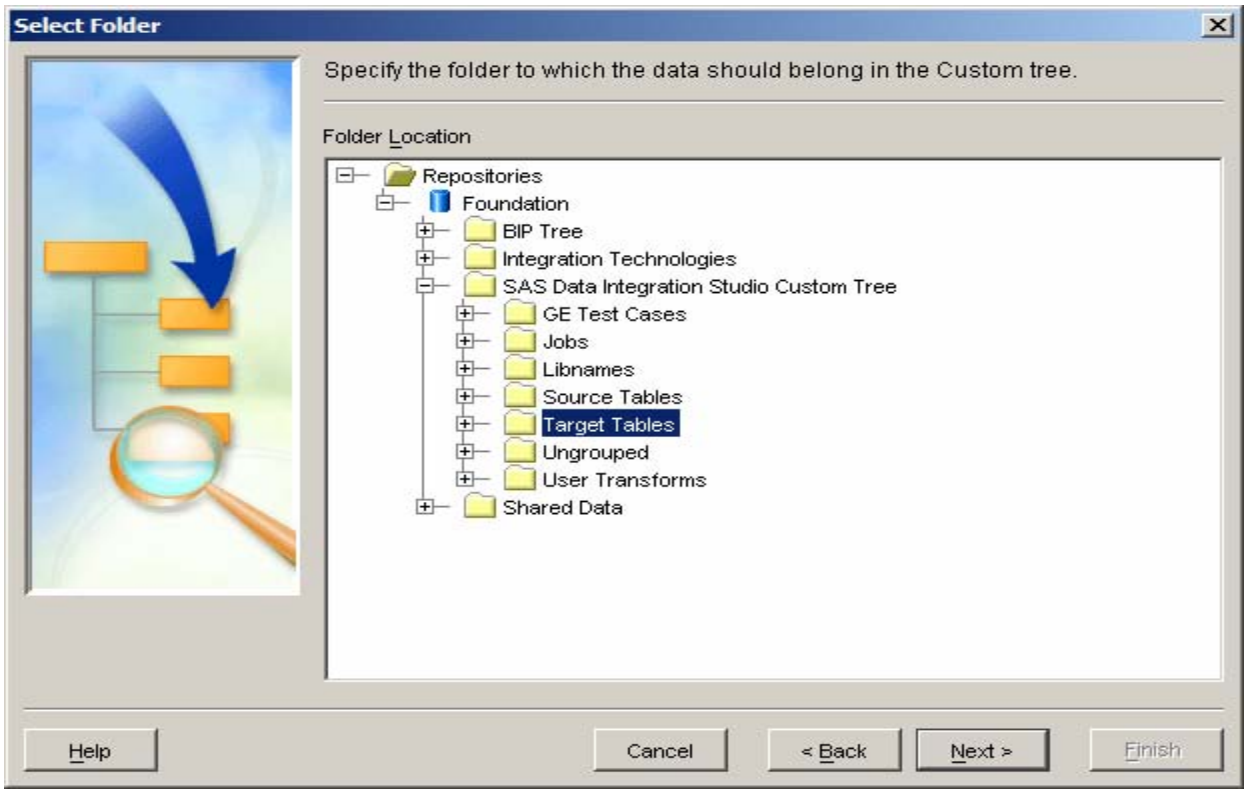


The next step provides the option of modifying the table's column information. If some of the columns needed do not exist in metadata they can be created manually in this step.



At this point all of the metadata about the table has been gathered, and the only item remaining is to select where the

metadata should be stored.



The final page in the wizard provides a summary listing very similar to the source designer for your review. After your target table metadata has been created you are free to use the data in a variety of ways. One of the most useful is Impact Analysis.

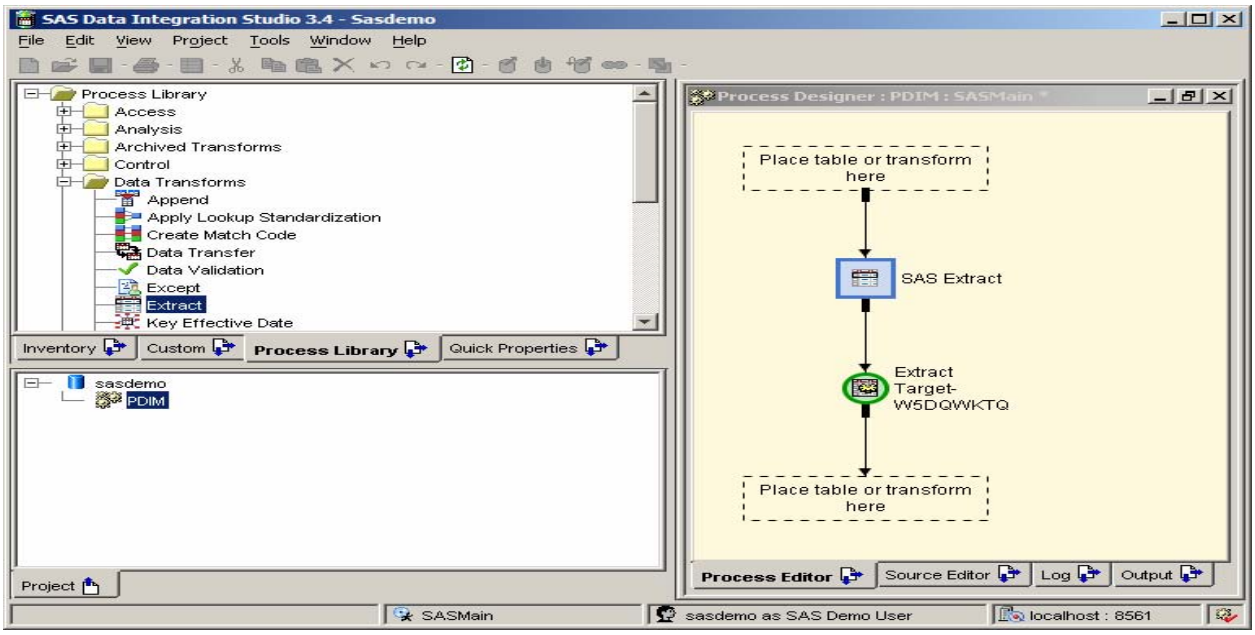
IMPACT ANALYSIS

Each transform (see below) takes a set of inputs and produces some output. As such, DI Studio provides a very useful utility for determining the impact of making changes to a table. To create the report you simply need to right click on the table and select "Impact Analysis…" A report is generated in a tree structure which allows you to view the lineage of the table. This lineage includes not only the jobs which use the table but the transforms linked to the table. The tree structure allows you to drive from the job to the transforms and the output tables of the transforms. The impact analysis report shows related jobs where the table is a source. There is also a "Reverse Impact Analysis" available which creates a report that lists the jobs where this table is the output. In addition to the report there is a separate tab which provides a graphical representation of the report.

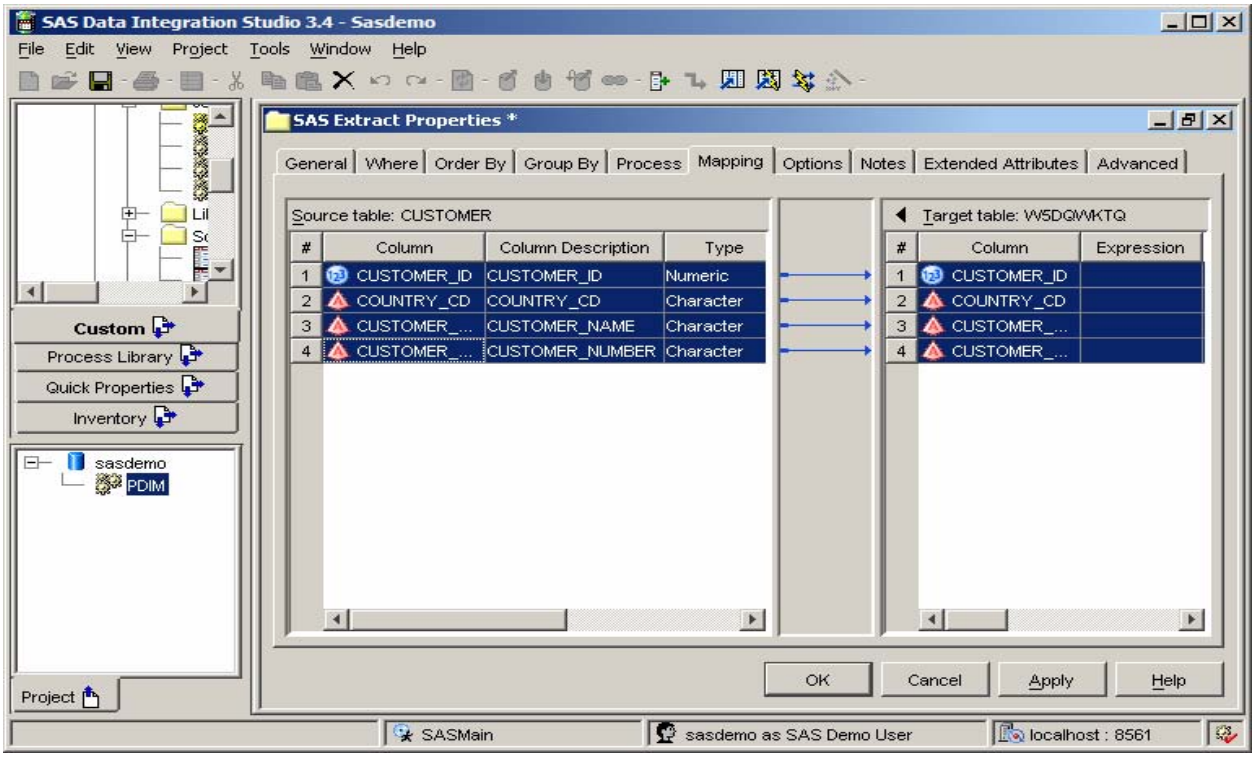
TRANSFORMATIONS

A transformation template is a process flow diagram that performs a certain task and includes input/output areas for you to drop metadata. DI Studio provides a number of ready made templates which perform many of the common tasks needed. In all the transforms there is the ability to override the generated code with custom code if needed. There is also a transform generator which gives you the ability create your own custom transforms to meet your needs. The transforms can be found on the "Process Library" tab in DI Studio. There are a number of groupings of transform templates in this tab. The "Data Transforms" grouping is where many of the transform templates are used for the example.

The diagram below shows a job with the "Extract" transform in it. To get the transform in the job you simply select the desired transform and drag it onto the job. The dashed line boxes signify the drop zones which is where tables or other transforms will be dropped to provide most of the info the transform needs to be complete.

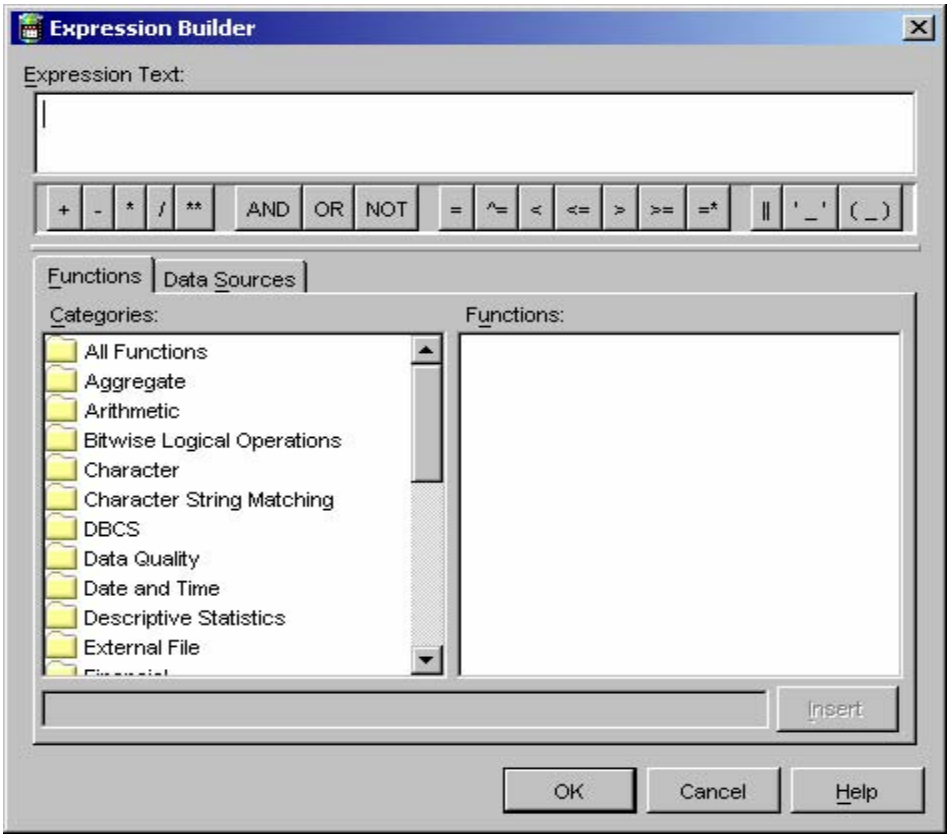


Double clicking on the transform will bring up the transform property window. The properties window contains numerous tabs, many of which are similar across transformation templates and some which are specific to that transform. The mapping tab is found in almost all templates and is one of the most important tabs. This is where you map the input columns to the output columns. To create a mapping you simply select a column on the left hand side by clicking on and holding down the mouse button then dragging the pointer to the column on the right hand side. If the source column names are the same as the target column names DI Studio provides the ability to automatically map these for you. Simply right click on the mappings tab and select the “Quick Propogate” option.



If there is a need to make changes or combine columns there is an expression box where this can be done. DI Studio helps with the process via an Expression builder. Double clicking in the expression box will cause a button to be

displayed which can then be clicked to get to the expression builder. There are two tabs in the expression builder; one which provides a comprehensive list of the available functions and the other a list of the available data. This process of mapping and building expressions will be used in almost all the transforms.



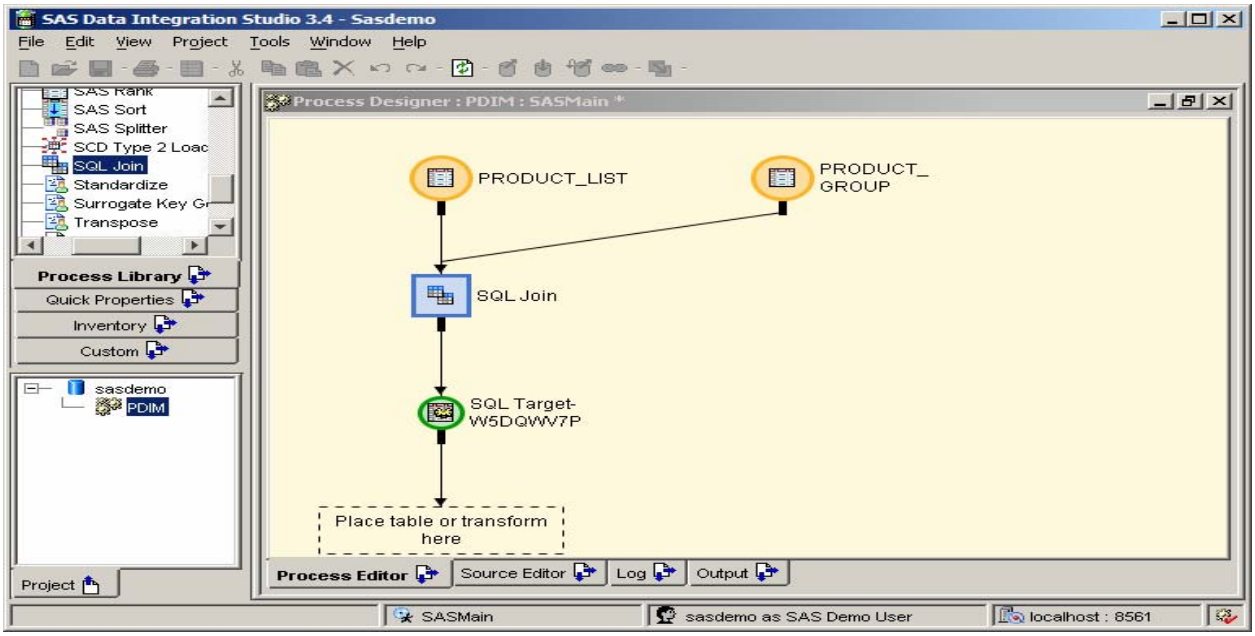
Another important aspect of working with transform templates is in choosing the best transform for the task needed. DI Studio comes preloaded with many transforms that perform a wide range of activities. Below is a list of transforms which should provide you with a good start in your DI Studio development.

- Extract – A simple template which provides functionality to filter, sort, and group data
- SQL Join – This transform requires two or more input tables. The user defines key relationships between the tables and how the joins will be performed.
- Loader – Used to load the data into a target table. The loader provides a flexible mechanism to load data; full refresh, append, update, etc. If the data is fully refreshed there is also the option of truncating the table or dropping and recreating it.
- User Written Code – This is a basic transform that doesn't generate any code but merely provides a template for hand written code.
- SCD Type2 Loader – This transform implements Ralph Kimball's Type 2 Slowly changing dimensions. The transform handles surrogate key generation, the ability to identify changes, and loading the data. Change tracking can be implemented in a number of ways; valid from and to dates, version numbers, or an indicator for the current record.
- Fact Table Lookup – Provides the ability to map the business keys in a fact table to their respective dimension surrogate keys. The transform includes the ability to perform actions if the key is not found.
- Splitter – Provides the ability to split a table into two sub-tables based on row separation or column separation.

- Sort – provides the ability to sort a table (Proc Sort)

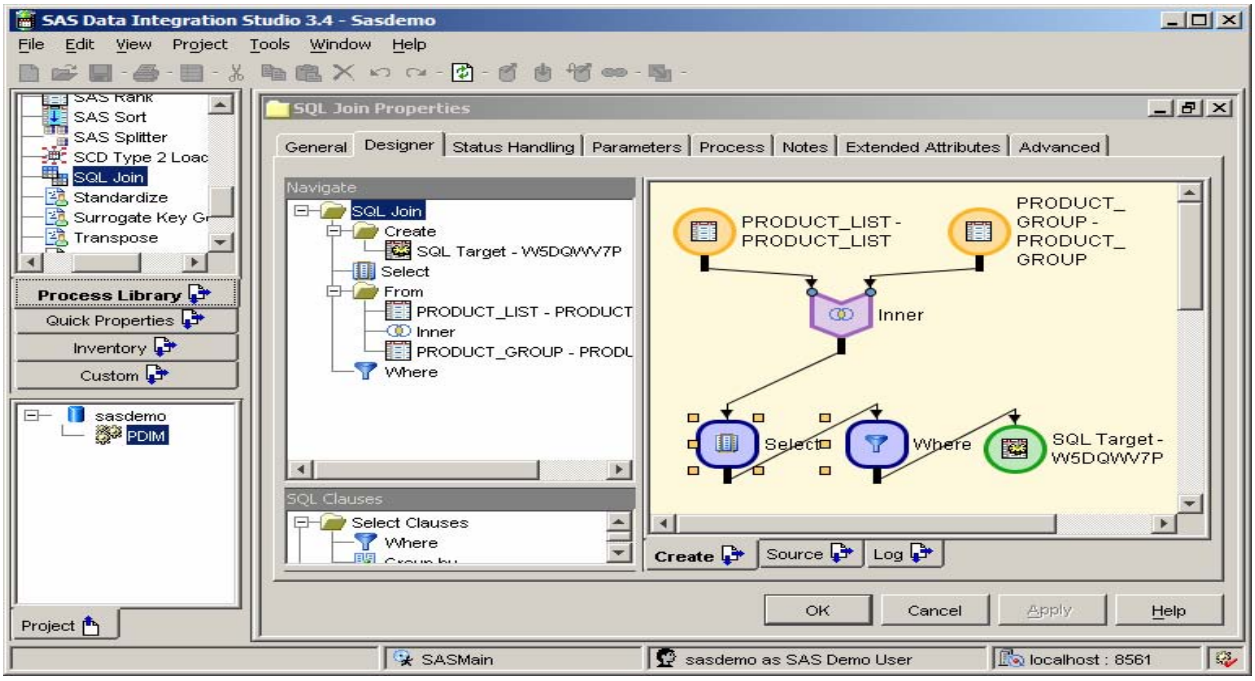
SQL JOIN

One of the most useful transformation types in the transform library is the SQL Join transform. This transform has been completely rewritten for version 3.4 of DI Studio, and is much more useful. The SQL Join step provides the ability to merge multiple input tables into a single table using SQL.



To add multiple input tables to the transform, drag-n-drop the input table onto the SQL Join node on the diagram Note, this is not entirely obvious, as the SQL Join transform starts with an empty box for the first input. It only took the author 2 hours to figure out that the 2nd and subsequent tables must be dropped on the actual icon. As an alternative, you can RMB on the SQL icon and select “Add Input”.

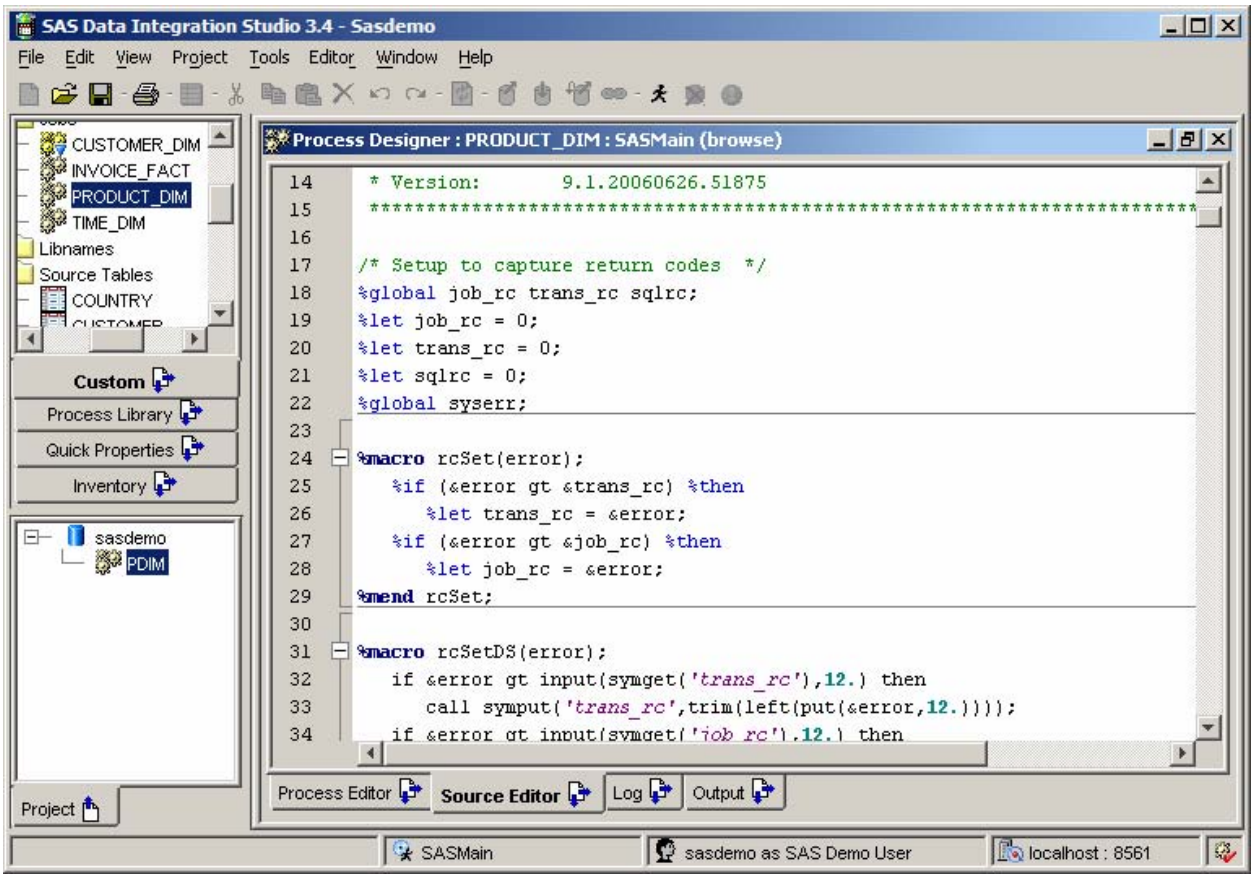
The main component of the SQL Join transform is the Designer tab found on the Properties screen (double click the SQL Join icon). The designer tab gives you complete control over how you will join and manage the multiple tables in the step.



All types of joins are supported (inner, outer, left, right, full outer) as well as the individual sub-steps of the join (where, having, order by, group by…). To view a sub-steps information, double-click the icon. To see the SQL step's code, select the Source tab in the designer.

TESTING

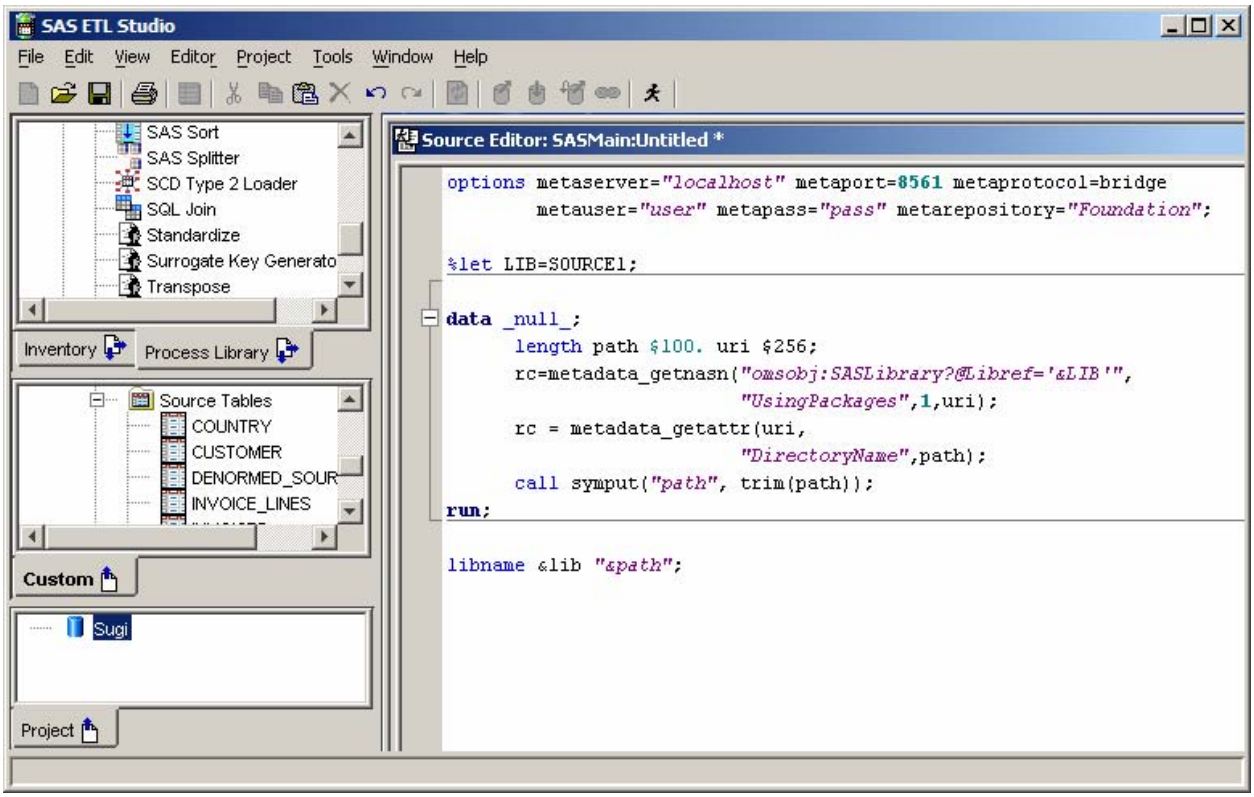
Testing your job involves generating source code and running it on one of the managed SAS servers. All SAS servers are defined in the metadata repository (see the SMC for more information). DI Studio uses the server metadata from the SMC to automatically submit source code to the correct server. In the application, you will see a Source Editor tab as part of the job. Selecting this tab will auto-generate the source code:



Once the source is generated, you can submit it by clicking the Submit Source button (the little running man on the toolbar). The code will be submitted to the SAS server and executed. When the job is complete then the log tab will be filled in with the resultant log. Any errors will be highlighted. Note, you can resubmit a job as many times as you want; however, sometimes the error in question will get SAS into a bad state. If this occurs, terminate the job window (click the X to close the window) and bring it back up. This will start a new SAS session.

If you would like to run arbitrary SAS code for testing purposes then you can use Ctrl-E to launch a source window. This window allows you to submit whatever you like to the backend SAS server. Note however, that libnames are automatically created when a job's source code is generated. If you need a libname, sometimes it is easier to use a User Written transform with an input. Alternatively, you can write data step code to read a libname's attributes from the metadata repository. For instance, code similar to the following will be required (see below).

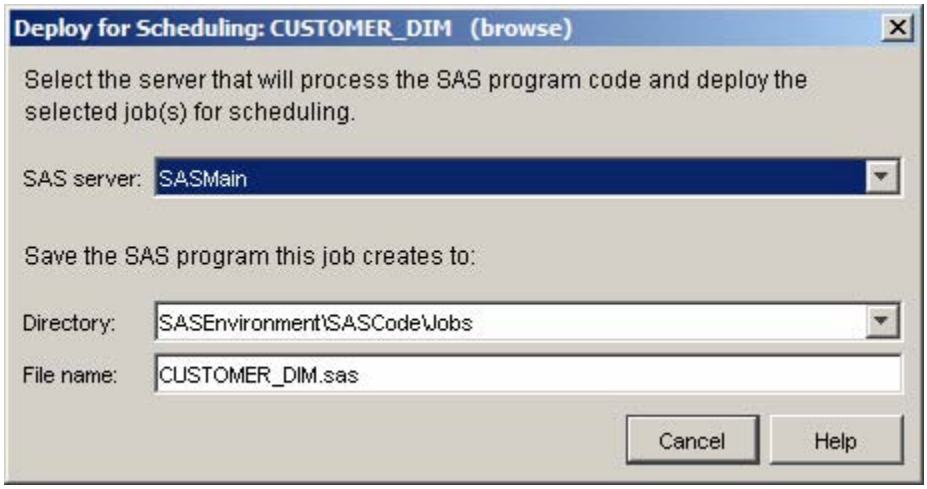
Version 9 also provides a new type of libname that can be used to reference SMC stored libnames. The keyword META assigns a libname that will first look at the metadata repository for the underlying libname. Information about the data sets stored in the library is passed through to the library with extra permissions information from the permissions model in SMC. This allows you to fully provide security information regarding tables. For more information see the documentation on the Libname Statement for the Metadata Engine.



You can also modify DI Studio generated code in the Source Editor tab. Any edits will be lost the next time the code is generated, so be sure to save them if you want to keep them. You can deploy and schedule a job upon completion of your testing using the SMC.

DEPLOYING AND SCHEDULING

Once a job is completed it must be deployed to make it available for scheduling. Deploying a job generates the code for the job and creates a SAS file which the scheduler will execute. To deploy the job simply right click on the job in Foundation and select "Deploy for Scheduling". The user is then prompted with a dialog that allows them to select the SAS Server, the deployment directory, and the SAS file name. The SAS Administrator will determine which SAS Server should be selected and the correct deployment directory. The default value for the file name is the name of the DI Studio job with a SAS extension but this can be overridden if necessary. A deployed job can be managed via the "Schedule Manager" plug-in in SAS Management Console.



CONCLUSION - THE INS AND OUTS

After using the tool for a while we have formed some opinions about how the tools functions (as expected). The following is a list of things that we found useful (the Ins) and a list of things that we think need some work (the Outs), and some tips that might help you on our way:

THE INS

- A visual representation of a job - The canvas rendering and options for layout work well. Being able to see the layout of your job is very helpful in determining what it does.
- Auto Map – The first time through, anyways. See the Outs.
- Job Import and Merge Wizard
- Job Export Wizard in release 3.4
- Job Status Manager
- Impact Analysis – in the authors opinion, one of the nicest features of DI Studio
- Pre and Post processing functionality for steps

THE OUTS

- Auto Map – after initial assignment, any change in mapping or inputs cascades, sometimes with undesirable results, down the rest of the transform chain
- Project Repository Corruption – DI Studio relies on the metadata repository for all of its storage. Make sure you have the latest fixes/release of SAS and DI Studio. Check in often.
- Unable to perform selective check-in – you must check in a whole repository. Again, check in often.
- Generated code is very complex. Even though SAS is generating it, you may still need a SAS programmer to work out any issues that may arise
- Large metadata repositories suffer from performance problems.

TIPS

- Turn off Auto Map (or you will be sad ☹). To do this RMB on each transform node and uncheck Automap. There needs to be a global setting for this, but there is not.
- To add multiple input data sets to a transform that can handle them, drag-n-drop the input data set onto the transform node.
- Save your project repository back to Foundation often, and back up (Export) your jobs frequently.
- Make sure you have all of the hot-fixes for DI Studio. The initial release of 3.4 had a bug where source code was generated out of order (nice!).
- If a Job's underlying SAS session seems to be hung or goobered, close the Job window and reopen it to reset it.
- Include macros by using either a custom transformation, or by using a Code node. There is no specific provision for macro inclusion in the tool
- Use Ctrl-E to bring up a adhoc source code window. Note, Ctrl-E does not clear the window like in DMS!

ACKNOWLEDGMENTS

The authors would like to thank the folks on the R&D DI Studio team for their patient explanations. We learned a lot from you!

CONTACT INFORMATION

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

Chris Olinger
d-Wise Technologies, Inc.
3115 Belspring Lane
Raleigh NC 27612
Work Phone: 919-696-3276
Fax: 919-510-8391
E-mail: colinger@d-wise.com
Web: http://www.d-wise.com

Tim Weeks
SAS Institute, Inc.
SAS Campus Drive
Cary NC 27513
Work Phone: 919-531-0406
Fax: 919-677-4444
E-mail: tim.weeks@sas.com
Web: http://www.sas.com