Understanding SAS/Warehouse Administrator®

Michael Davis, Bassett Consulting Services, North Haven, Connecticut

ABSTRACT

Some firms have looked at SAS/Warehouse Administrator® and decided to continue developing applications in the traditional way. Why? Reasons include high investment cost, difficulty of incorporating legacy code, and the awkwardness of using terminal emulators when running SAS/Warehouse Administrator on a server located in a "glass room" or other remote location.

However, if one can overcome these objections and let SAS/Warehouse Administrator write the required SAS® code, the advantages of metadata take over. These advantages include easier maintenance and more rapid development of new data warehouse applications. Another advantage includes quickly determining the impact of changing columns and rows. Last, SAS/Warehouse Administrator automatically publishes HTML documentation and process diagrams.

This paper will illustrate how a data mart is modeled in SAS/Warehouse Administrator, drawing upon a prototype that the author recently created. The example will highlight how conditional processing can be accommodated by CASE expressions. A technique to work around the limitations of terminal emulation will also be demonstrated.

WHAT IS DATA WAREHOUSING ?

Before one can make the case for selecting SAS/Warehouse Administrator as the tool to use when creating a data warehouse or data mart, it is important to define what data warehousing means.

The author defines data warehousing as the process of making operational data available to decision support applications, such as SAS. Data warehousing involves extracting, transforming, joining, sorting, summarizing, and consolidating operational data.

INTRODUCTION TO DATA WAREHOUSING

The skeptical reader might ask, "Why undertake this activity?" To address the concerns of such readers, one might offer the following common computer programming [SAS] activities that are performed in the cause of data warehousing:

- *Raw operational data is filtered into a sub-set* to remove columns and rows that are not required for typical decision support activities. Filtering raw data down to the essential items can improve the speed of subsequent decision support activities.

- *"Header" and "Detail" files need to be joined*. In operational data files, common information such as a client's physical address is removed from the transactional detail files and placed into a header file to save disk space. To analyze the transactions, it is often necessary to re-join the header information with the transactions in the detail file.

- *Tables need to be sorted and indexed*. Operational data is often sorted by the keys necessary to quickly find a customer transaction. It would be a lucky accident to find that this order is also the best sort sequence to support decision support reporting. Because tables often need to be accessed by multiple keys, it is often necessary to create multiple indexes to promote efficient information retrieval.

- *Tables need to be summarized*. Decision support activities often require summarized data, collapsed by the analysis categories. Rather than going through the effort and expense of summarizing the detailed information each time a report is run, a better strategy is to pre-summarize the detailed tables once into the summaries that may be needed.

- *OLAP*. When multiple summaries of the same detail information is required, disk space and retrieval performance can often be improved by storing the summarized information in OLAP (On-Line Analytical Processing) structures, often known as "cubes".

- *Standardize code schemes*. As an example, a customer's sex might be denoted in one table as either "M" or "F". In another table, the information may be stored as 0 or 1. To facilitate enterprise-wide reporting, it is desirable to transform disparate code schemes to common ones.

- *Cleanse "dirty" data*. Consider the previous example. If the customer (patient) is coded as "M" (male) and pregnant, then it appears that an effort to correct this and other logical inconsistencies should be mounted. Edit-check programs can identify elements that require cleansing and can perhaps correct some errors without manual intervention.

- *Standardize physical file structures*. Some data may be in flat files, some in SAS data sets, and others in third-party data base management systems (DBMSs) such as Oracle®, Microsoft® SQL Server™, and DB2. To facilitate reporting, the data should be transformed into tables of same physical format. If the reporting is to be done with SAS tools, then the data should be stored in SAS tables, MDDB cubes, or views created by SAS/ACCESS®.

BENEFITS OF DATA WAREHOUSING

At this point, the skeptical reader might exclaim, "I can see the value of data warehousing but how do I justify the cost and effort to my management?" Here are some common benefits that data warehousing can yield:

- *Reduce intra-organization discrepancies*. When each department or division undertakes the preparation of raw data for reporting, different assumptions and techniques can yield different results. This can lead to more effort being spent on reconciling the differences than on what the results mean.

- *Reproducible results*. Operational data often changes. If one runs the same report later, the results may differ from the first run. When the data source for a report is a data warehouse or mart table that represents a "snapshot" taken at a specified interval, the report's users can count on consistent results.

- *Document data repository*. Much effort is consumed in answering questions such as "Where does that number come from?" and "What does that code represent?" One common benefit of data warehousing is that the process and resulting data stores of the data warehouse and marts are documented for the benefit of the potential users.

- *Improve performance of operational systems*. Transaction systems are often designed to give best performance when a few records are to be retrieved. By contrast, decision support applications typically read entire tables. When transaction and decision support systems share the same data sources, performance of the transaction systems can suffer. A better solution may be to create or update data warehouse and mart tables from the operational systems during off-peak hours.

- *Save human resources*. In organizations where data warehousing is not well organized, multiple persons often duplicate efforts to transform operational data for reporting. Some of these persons may not have the appropriate skills or tools to perform this task. One of the benefits to data warehousing is to save human effort and costs in creating and maintaining data warehouses and data marts.

WHY SAS/WAREHOUSE ADMINISTRATOR ?

Some readers might exclaim at this point, "Yes, we see the value to data warehousing. But why should we try to convince management to license yet another SAS product? Can't we do data warehousing with Base SAS?" Of course organizations can create data warehouses and data marts with Base SAS and other tools.

However, it is this author's proposition that when one considers the total cost and effort required to create, maintain, schedule, and document data warehouses and data marts, licensing SAS/Warehouse Administrator may be the least expensive alternative. Consider the following benefits that may be gained by using this product:

- "point and click" interface

- ability to accumulate, maintain, and report on the warehouse's metadata

- control processes across multiple platforms

- changes are automatically posted to generated SAS code

- process flowcharts can be generated

- HTML document can be generated and posted to a web server

- Dependent job scheduling and load-sharing can be accomplished via the LSF JobScheduler

- Process libraries and other features are available to structure the warehousing process

WHAT IS METADATA AND WHY IS IT NEEDED ?

Metadata is information that defines sources, data stores, code libraries, and other resources. It is used to write the actual SAS code. Technical metadata defines where the data lives and how to access it. Business metadata defines what the data means and who is responsible for it.

Perhaps the major advantage of using SAS/Warehouse Administrator is that it facilitates the creation and maintenance of metadata. Consider the following example. It is decided to change the logic used to transform a column of intermediate information? How do we find all of the places affected by the change and make sure that they use the new logic?

Without a tool such as SAS/Warehouse Administrator, making changes to an existing data warehouse or mart can be a nightmare. Metadata gives us a single point of control, even when warehousing occurs across multiple computer platforms.

SAS/Warehouse Administrator facilitates changes to programs that create and maintain data warehouses and marts because it actually generates the SAS code to be run. It also provides tools to search, report, and document the metadata. Finally, SAS/Warehouse Administrator can import and export metadata to other applications. This opens the possibility of using additional tools to create and maintain a data warehouse or mart.

DATA WAREHOUSING VOCABULARY

The term “data warehouse” is commonly used to describe all outputs of data warehousing. However, it is the author’s conclusion that many repositories created by data warehousing are more accurately described as “data marts”. Data marts are distinguished from data warehouses in that they are organized to support a specialized, specific application and a finite set of reports.

The acronym “ETL” stands for extract, transform, and load. ETL processes represent the major activity associated with data warehousing and the use of SAS/Warehouse Administrator.

The SAS web site lists other data warehousing vocabulary that may be helpful to the uninitiated. It can be found on the SAS web site, in the Data Warehousing Community at:

http://www.sas.com/rnd/warehousing/glossary.html

Another feature of the Data Warehousing Community section of the SAS web site that is well worth exploring is the “Getting Started with SAS/Warehouse Administrator”. Also supplied with the software, it can be found at:

http://www.sas.com/service/tutorials/v8/warehous/index.html

This tutorial is invaluable to those attempting to set up their first project in SAS/Warehouse Administrator.

STARTING SAS/WAREHOUSE ADMINISTRATOR

The current version of SAS/Warehouse Administrator is run within a SAS session as a SAS desktop application. The SAS desktop is a graphical interface to tools and files. It is supplied with Base SAS. However, the SAS desktop is more commonly used to access features of SAS/EIS and other SAS products.

To start SAS/Warehouse Administrator, one can select -> Solutions -> Development and Programming -> Warehouse Administrator. However, the author finds it more convenient to issue the command “DW” from the command bar. This will open up a window similar to the one shown in Figure 1:

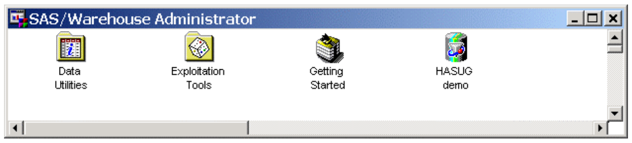


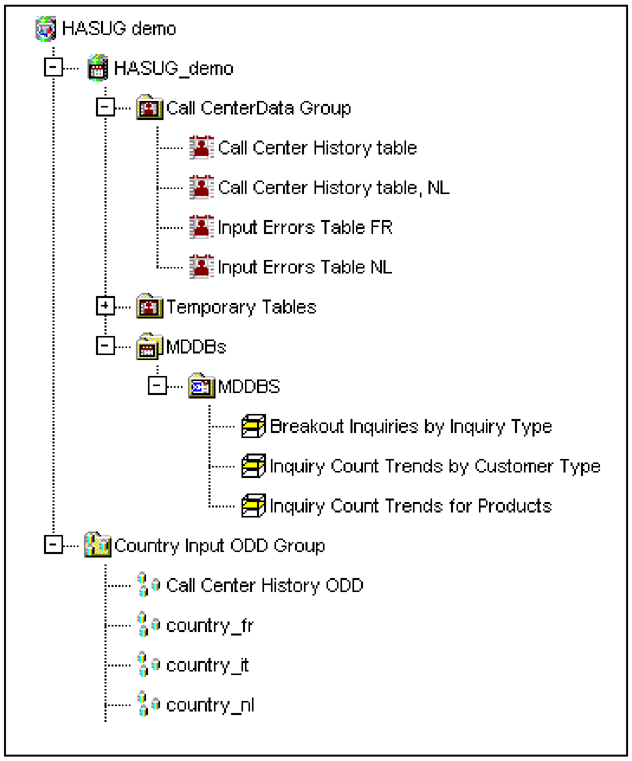
Figure 1

To open an existing warehouse environment, one merely double-clicks on the icon representing that environment. To create a new warehouse environment, right-click on white space within the SAS/Warehouse Administrator window.

Environment Hierarchy

One of the confusing aspects of SAS/Warehouse Administrator that confronts new users is the hierarchy of warehouse elements. The hierarchy is illustrated in Figure 2, shown on the next page. The following limited hierarchy description may help those beginning to use this product.

Within a typical warehouse environment, there are usually Data Warehouses and Operational Data Definition Groups. Data Warehouses are further organized into Subjects, which may contain Data Groups, Infomarts, and OLAP Groups. Operational Data Groups Definition Groups include one or more Operational Data Definitions. Figure 2 shows the hierarchy of groups and data stores when our "demo" environment is opened:



```
HASUG demo
⊟ HASUG_demo
    ⊟ Call CenterData Group
        Call Center History table
        Call Center History table, NL
        Input Errors Table FR
        Input Errors Table NL
    ⊞ Temporary Tables
    ⊟ MDDBs
        ⊟ MDDBS
            Breakout Inquiries by Inquiry Type
            Inquiry Count Trends by Customer Type
            Inquiry Count Trends for Products
⊟ Country Input ODD Group
    Call Center History ODD
    country_fr
    country_it
    country_nl
```

Figure 2

This demo environment was created to illustrate how data from telephone calls made to customer service centers in multiple countries might be periodically consolidated and summarized.

In our demo environment, we see the HASUG demo warehouse environment icon at the top of the hierarchy. Next in the hierarchy, we see HASUG_demo data warehouse icon and the Country Input ODD (Operational Data Definition) Group icon.

Under the HASUG_demo data warehouse icon is the Call Center Data Group, the Temporary Files data group, and the MDDBs subject. In Figure 2, the icons for the history tables and input errors tables are shown under the Call Center Data Group.

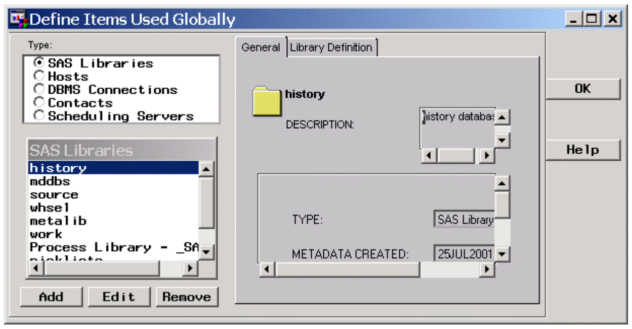
Under the MDDBs subject icon, there is a single MDDBS OLAP group. Under this group are the three MDDB OLAP cubes created by the demo environment.

The Count Input ODD Group defines all of the input sources to our warehouse. In addition to the files received periodically from three countries (France, Italy, and Netherlands), the Call Center History table also appears. This icon refers to the same physical table that the Call Center History data table. This is because the inputs to warehouse processes must be defined as ODDs and output tables must defined as data tables.

GLOBAL METADATA

In our demo, there is just a single warehouse. However, in practice, we may wish to create multiple warehouses. It would be a nuisance at best if we had to define global parameters for each warehouse. So SAS/Warehouse Administrator allows us to define in one place all of the metadata that may be shared across multiple warehouses in the same environment.

To get to the global metadata in SAS/Warehouse Administrator, one selects -> File -> Setup… A window similar to the one illustrated in Figure 3 appears:



Define Items Used Globally

Type:
(● SAS Libraries) ○ Hosts ○ DBMS Connections ○ Contacts ○ Scheduling Servers

SAS Libraries: history (selected), mddbs, source, whse1, metalib, work, Process Library - _SA, picklists

General | Library Definition

history
DESCRIPTION: history databas
TYPE: SAS Library
METADATA CREATED: 25JUL2001

Figure 3

One can create, examine update, and delete different categories of metadata by selecting a radio station in the Type window, selecting the metadata item in the window below it (SAS Libraries as shown in Figure 3), and clicking on the Add, Edit, or Remove buttons.

SAS library metadata is largely self-explanatory. However, one of the author's tricks is to use SAS/ACCESS Libname engines to define data sources in DBMSs as SAS libraries rather than DBMS Connections. This trick seems to work better when the DBMS options are embedded in the Path text box instead of the Options text box on the Details tab.

In our demo, only the computer on which SAS/Warehouse Administrator is defined as a host computer. However, in a distributed computing environment, remote hosts can be defined to and can be controlled by SAS/Warehouse Administrator.

All elements of a data warehouse have an owner and an administrator as attributes. The contact information for all individuals who serve in these roles is defined in a single place. This makes updating this information much more convenient.

Last, the information about the scheduling servers is entered as global metadata. SAS/Warehouse Administrator allows users to define CRON, AT, and null scheduling servers. The null scheduling server writes a file that is used by the LSF JobScheduler and other third-party scheduling servers.

TYPICAL SAS LIBRARIES

The libraries (librefs) that should be entered into the SAS/Warehouse Administrator will vary with each project. Some of the libraries defined as global metadata will be assigned by SAS/Warehouse Administrator. Other libraries may be assigned externally, either when the SAS session is started, or as part of user-written SAS code.

The following libraries are typically assigned as part of a warehouse environment's metadata:

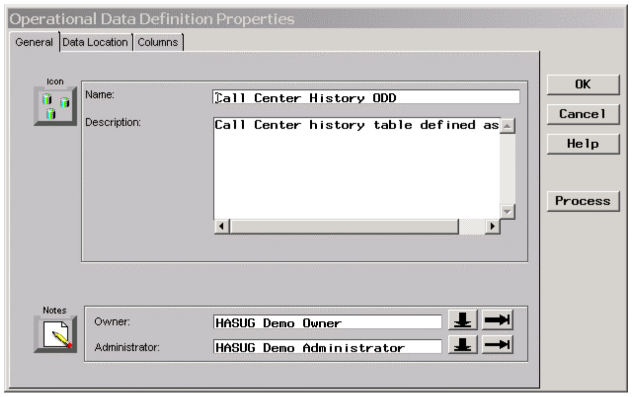
- DBMS engine librefs
- Detail Data
- Source Code
- Metalib (_DWMD)
- Process Library (_SASWA)
- Warehouses

The _DWMD and _SASWA are required by SAS/Warehouse Administrator. The requirements of add-in tools make it a good idea to assign the _SASWA externally through the autoexec.sas program.

The author often defines the Work libref as part of the metadata so it can be used when defining temporary tables that should disappear when the SAS session ends. If MDDBs are to be used outside of SAS/Warehouse Administrator, such as with AppDev Studio™ or WebHound™ software, then it may be useful to define an MDDB libref in the global metadata and assign it through the autoexec.sas program.

OPERATIONAL DATA DEFINITIONS

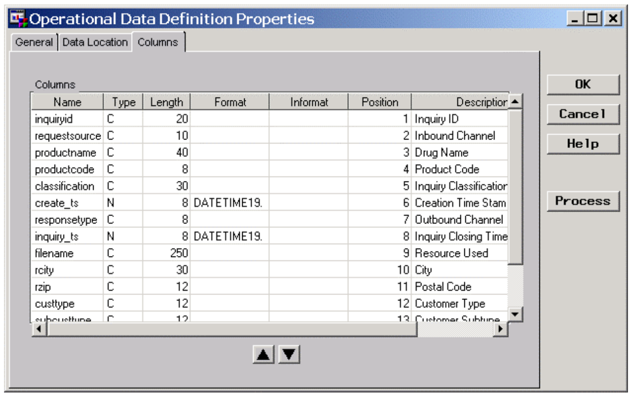
Operational Data Definitions are metadata records that provide the instructions to access data sources. Figure 4 illustrates an Operational Data Definition Properties Window.



Name:	Call Center History ODD
Description:	Call Center history table defined as
Owner:	HASUG Demo Owner
Administrator:	HASUG Demo Administrator

Figure 4

The General tab allows one to enter a description and indicate the table's owner and administrator. The Data tab allows one to specify the host, library, and table name. The Columns tab, illustrated in Figure 5, shows the type information about the table's variables (columns) that one would see in a "contents" listing.



Columns

Name	Type	Length	Format	Informat	Position	Descriptior
inquiryid	C	20			1	Inquiry ID
requestsource	C	10			2	Inbound Channel
productname	C	40			3	Drug Name
productcode	C	8			4	Product Code
classification	C	30			5	Inquiry Classificatior
create_ts	N	8	DATETIME19.		6	Creation Time Stam
responsetype	C	8			7	Outbound Channel
inquiry_ts	N	8	DATETIME19.		8	Inquiry Closing Time
filename	C	250			9	Resource Used
rcity	C	30			10	City
rzip	C	12			11	Postal Code
custtype	C	12			12	Customer Type
subcusttype	C	12			13	Customer Subtype

Figure 5

Two tips can be shared about the Columns tab. It can take a bit of time to key in the required information for a new table. If a similar table already exists or can be generated by running some legacy SAS code, it is much faster and easier to "import" the required information from that table. Also, after moving rows up or down with the arrowhead buttons at the bottom of the window, right-click on a row and select Save Order to retain the new order after the properties window is closed.

At this point, it might be a good idea to define any additional ODDs required. After all the ODDs have been defined, the next logical step is usually to define the required output structures.

These include data tables, MDDB cubes, and other output files. These typically include flat files, comma-separated value (CSV), and various Output Delivery System (ODS) destinations. Then it is time to start defining the transformation of input tables into target outputs from the Process Editor.

USING THE PROCESS EDITOR

The Process Editor is used to manage jobs, job flows, and process flows. These properties must be defined in order for SAS/Warehouse Administrator to generate the source code for the transformation jobs.

There are multiple ways to bring up the Process Editor. One method is to select Tools -> Process Editor from the pull-down menus. Figure 6 shows a sample Process Editor window.

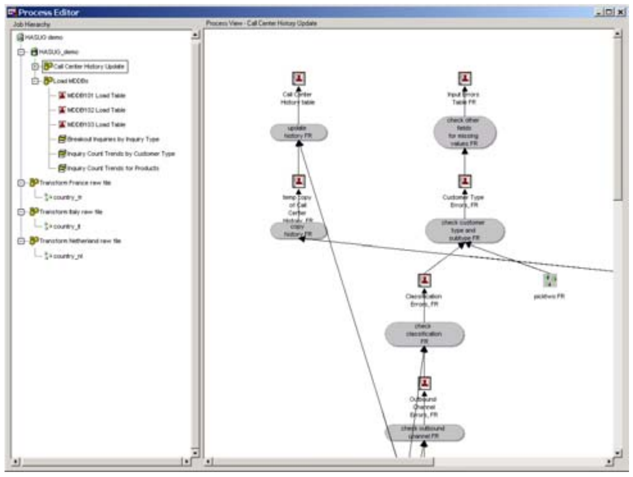
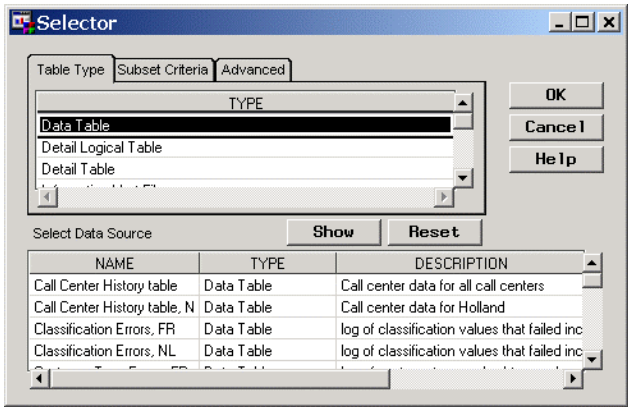


Figure 6

The Process Editor window consists of two panes. In Figure 6, the left pane shows the Job Hierarchy after it has been partly expanded. Under the jobs are the output tables and files produced by the jobs.

The right pane shows the part of process flow for the job, output table, or file currently selected in the Job Hierarchy pane. The direction of flow is from bottom to top, left to right.

The author recommends adding the process output(s) first. This is done by right-clicking in the Process View pane and selecting Add Output Table… This brings up the selector shown in Figure 7. Select the category of output table to be added and click on the Show button to display the output tables available to be added.



TYPE
Data Table
Detail Logical Table
Detail Table

Select Data Source

NAME	TYPE	DESCRIPTION
Call Center History table	Data Table	Call center data for all call centers
Call Center History table, N	Data Table	Call center data for Holland
Classification Errors, FR	Data Table	log of classification values that failed inc
Classification Errors, NL	Data Table	log of classification values that failed inc

Figure 7

Next, for each output table, define the input data source(s). This is done from a selector similar to the one shown in Figure 7. Again the same physical table can be an output table and an input table within the same process flow.

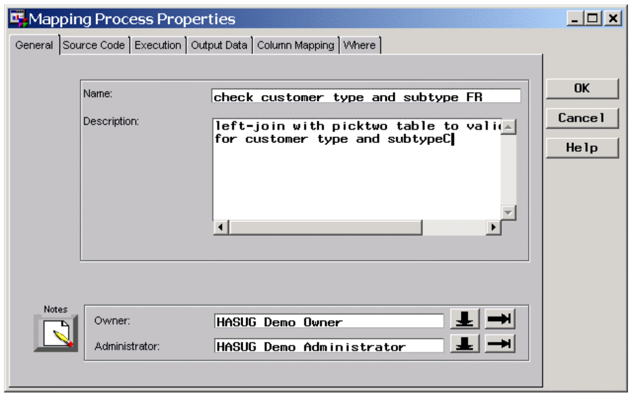
MAPPING STEPS

Experienced SAS users may ask, “How do I transform the information contained in an ODD into an output table or MDDB cube?” “How do I embed these transformations within the metadata?” This is done through mapping steps.

As one might anticipate, mapping steps define how columns and rows from the input tables are mapped to output tables or MDDB cubes. Mapping steps can specify one-to-one, one-to-many, or many-to-one mappings. SAS/Warehouse Administrator uses the metadata in the mapping steps to generate PROC SQL code to effect the transformations. However, add-in-tools can be used to customize the behavior of the mapping steps.

An example of the dialog box that sets the mapping step is shown in Figure 8 on the following page. The contents of the General tab are displayed. On this tab, and through SAS/Warehouse Administrator, one can add annotations by clicking on the Notes button, which brings up a notepad window. The notes are saved in a catalog source entry.

An example of the Source Code tab is shown in Figure 9 on the following page. This tab allows one to select whether the SAS transformation code is generated by SAS/Warehouse Administrator or is written by the user.



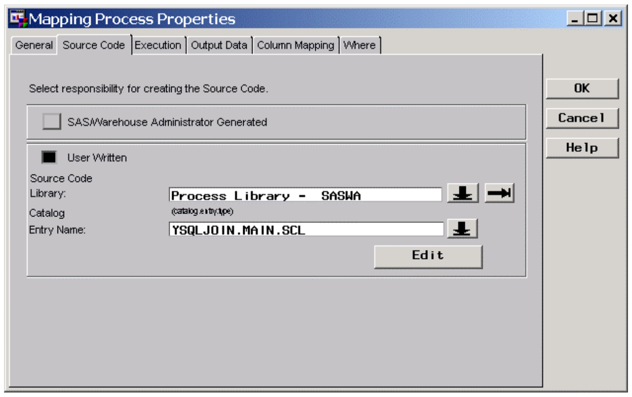
Name: check customer type and subtype FR

Description: left-join with picktwo table to vali
for customer type and subtypeC

Owner: HASUG Demo Owner

Administrator: HASUG Demo Administrator

Figure 8



Select responsibility for creating the Source Code.

☐ SAS/Warehouse Administrator Generated

■ User Written

Source Code Library: Process Library - SASWA

(catalog.entry.type)

Catalog Entry Name: YSQLJOIN.MAIN.SCL

Figure 9

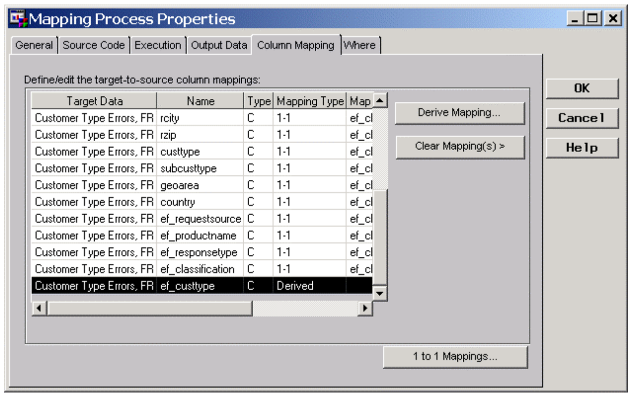
In mapping step illustrated in Figure 9, a two input table SQL join has been specified. When an add-in-tool has been specified, the Source Code Library selector will show "Process Library – SASWA" and the Catalog Entry Name selector will show the name of the catalog entry of the add-in tool.

The Execution tab specifies the computer on which the process is to execute. The Output Data tab specifies the location of the output table. An example of the Column Mapping table is shown in Figure 10.

One advanced use of the Output Data tab is to specify which rows are written to each of multiple output tables. This is done by first selecting the appropriate target table by clicking on the down arrow. Then while the desired target table is displayed, click on the Generation Options button. Then on the Row Selection tab, specify "Row Selection Conditions" or "User Defined Statements" to direct the rows to be output to the target table.

Mapping can be either 1 to 1 (1:1) or derived. If any of the column names are shared between the input and output tables, clicking on the button labeled "1 to 1 Mappings…" automatically sets mapping relationships for those columns whose names match.

While mapping relationships are often 1:1, SAS veterans will want to know how they can embed conditional mapping assignments. Those familiar with SQL (Structured Query Language) will recognize the solution, which are CASE expressions.



Define/edit the target-to-source column mappings:

Target Data	Name	Type	Mapping Type	Map
Customer Type Errors, FR	rcity	C	1-1	ef_cl
Customer Type Errors, FR	rzip	C	1-1	ef_cl
Customer Type Errors, FR	custtype	C	1-1	ef_cl
Customer Type Errors, FR	subcusttype	C	1-1	ef_cl
Customer Type Errors, FR	geoarea	C	1-1	ef_cl
Customer Type Errors, FR	country	C	1-1	ef_cl
Customer Type Errors, FR	ef_requestsource	C	1-1	ef_cl
Customer Type Errors, FR	ef_productname	C	1-1	ef_cl
Customer Type Errors, FR	ef_responsetype	C	1-1	ef_cl
Customer Type Errors, FR	ef_classification	C	1-1	ef_cl
Customer Type Errors, FR	ef_custtype	C	Derived	

Figure 10

Case expressions follow the format CASE… WHEN… ELSE… END. Consider the following example used to flag missing values.

```
case when picktwo_fr.= '' then 'X' else '' end
```

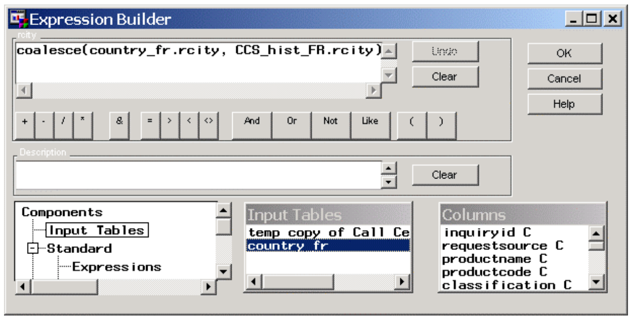
One derived mapping requirement that tested the author's creativity was how to code a "left" join. Left joins are often required when updating a master table. They are required because we only want to replace (update) information in the master table when a valid transaction has occurred.

The trick to accomplishing this feat is the Coalesce function. Consider the following example:

```
        coalesce(fr.city, history.city)
```

This expression replaces the value in the master (history) table for city with the value of city in the transaction (country) table only when city is a non-missing value in the transaction table (and when the WHERE keys match).

As noted earlier, one of the advantages of using SAS/Warehouse Administrator is that many expressions can be built using a "point and click" interface in lieu of typing, illustrated by Figure 11 on the following page.


Expression Builder

rcity

```
coalesce(country_fr.rcity, CCS_hist_FR.rcity)
```

Input Tables: temp copy of Call Ce, country fr

Columns: inquiryid C, requestsource C, productname C, productcode C, classification C

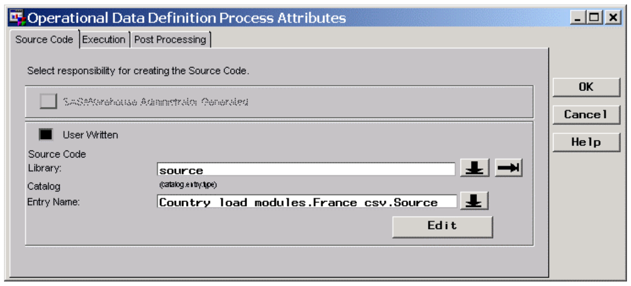
Figure 11

While almost everyone appreciates the reduction in typing offered by "point and click" interfaces, the reduction of the potential for typing errors is probably their greatest advantage.

The last feature of mapping steps is the specification of filtering on the WHERE tab. WHERE expressions may also be built in a "point and click" fashion in the Expression Builder. Use the WHERE tab to specify the merge keys when constructing a join process and to set filtering when sub-setting a table.

LOAD STEPS

Load steps are where one can place user-written code to override WA-generated SAS load code. To specify a load step, right-click on a output table or file in the Process View pane in the Process Editor and select Edit Load Step. The dialog box similar to the one illustrated in Figure 12 should appear.


Operational Data Definition Process Attributes

Select responsibility for creating the Source Code.

SAS/Warehouse Administrator Generated

■ User Written

Source Code Library: source

Catalog (catalog.entry.type) Entry Name: Country load modules.France csv.Source

Figure 12

Select "User Written" and specify the catalog source entry that contains the load code. Click on the Edit button to create or modify the load step.

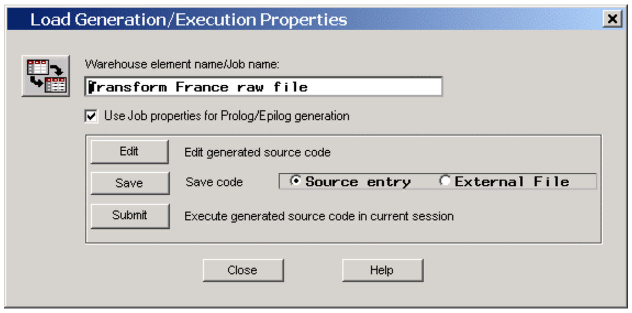
While there are many valid occasions where a user-written load step must be specified, there is a temptation to supply user-written load steps when the same result could be accomplished by the appropriate specifications to the warehouse metadata. This temptation should be strenuously avoided!

User-written code becomes a "black box". If other parts of the warehouse are changed or updated, the user-written load step will not automatically reflect those changes. The goal in using SAS/Warehouse Administrator is to model the process within the metadata and let the SAS/Warehouse Administrator generate the SAS code.

One last tip about load steps is not to forget to specify the host on which the load step is to run. It is very easy to forget this requirement.

EXECUTING THE JOB

There are two different ways to directly execute jobs entered into SAS/Warehouse Administrator. From the Job Hierarchy pane in the Process Editor, right-click on the job to be run and select Run… A dialog box similar to Figure 13 will appear.


Load Generation/Execution Properties

Warehouse element name/Job name: Transform France raw file

[✓] Use Job properties for Prolog/Epilog generation

Edit generated source code

Save code (● Source entry ○ External File)

Execute generated source code in current session

Figure 13

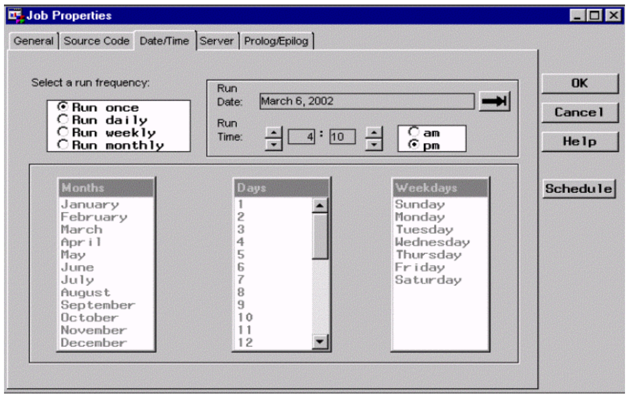
Click on the Edit button to generate the SAS code for the job in a Preview window and to edit it before submitting it. This is similar to selecting "View Code >" when right-clicking on a job.

Click on the Save button to generate the code associated with a job and save it to a catalog source entry or external file. This feature is very useful when the job is large and static. Once generated, an external job scheduler can launch the job. The Submit button causes a job to be generated and executed directly.

To schedule a job through SAS/Warehouse Administrator, it is necessary to set up a scheduling server as part of the global metadata and a job information library. If only a single host computer is used to run jobs, an ordinary libref allocation for the job information library will suffice. However, if multiple hosts run jobs, then SAS/SHARE should be used.

SCHEDULING JOBS

SAS/Warehouse Administrator can natively send jobs to CRON (Unix hosts) and AT (Windows hosts). The null scheduler generates a "stub" file that external job schedulers can read for scheduling information. To use the null scheduler, right-click on a job and select Properties. On the Date/Time tab, as illustrated by Figure 14, select when the job is to be run. Then on the Server tab, specify the null scheduler server.



Select a run frequency:
(● Run once) ○ Run daily ○ Run weekly ○ Run monthly

Run Date: March 6, 2002

Run Time: 4 : 10 ○ am (● pm)

Months	Days	Weekdays
January	1	Sunday
February	2	Monday
March	3	Tuesday
April	4	Wednesday
May	5	Thursday
June	6	Friday
July	7	Saturday
August	8	
September	9	
October	10	
November	11	
December	12	

Figure 14

One external scheduler that takes advantage of the null scheduler feature via an add-in tool is the LSF JobScheduler. LSF JobScheduler is a product of Platform Computing. An OEM license for LSF JobScheduler is supplied as part of SAS/Warehouse Administrator. To use LSF JobScheduler, it is necessary to request software keys from Platform Computing. LSF JobScheduler should be considered when dependent job scheduling or load-sharing is desired.

ADD-IN TOOLS

Add-in tools are programs written by the SAS warehouse developers (or users) to extend the functionality of SAS/Warehouse Administrator. They are installed on top of SAS/Warehouse Administrator and used to help load external data, model processes, schedule jobs, and to analyze, search, and report on metadata.

Add-in tools are usually accessed by right-clicking on an item in the Process Editor and selecting Add-Ins… The link to the information on add-in tools on the SAS web site is:

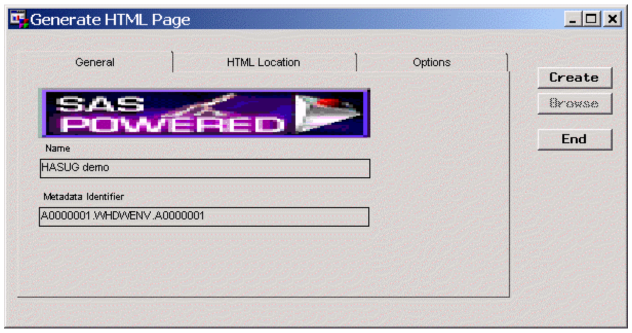
http://www.sas.com/rnd/warehousing/wa/addins.html

The list of add-in tools changes periodically and new versions of existing tools are often available for download. The application interface to SAS/Warehouse Administrator is documented so one can create their own add-in tools if they can code in SAS Component Language (SCL).

It is the author's understanding the add-in tools will disappear in a future version of SAS/Warehouse Administrator although the functionality that they provide should remain.

GENERATING HTML DOCUMENTATION

One of the author's favorite add-in tools is the one that automatically generates HTML documentation. To bring up this particular add-in, get out of the Process Editor and select Tools -> Add-Ins -> Publish metadata to HTML page from the pull-down menu. A dialog box similar to the one illustrated in Figure 15 should appear.

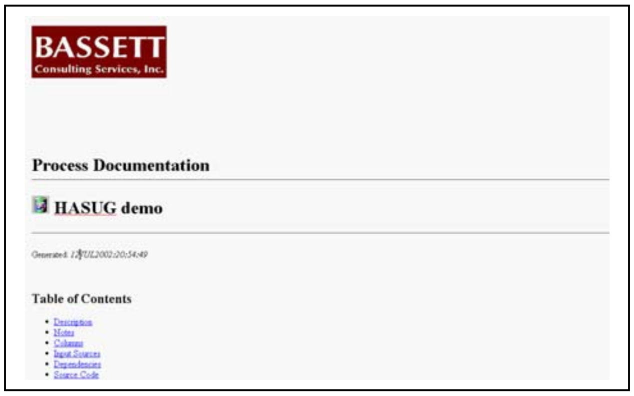


Name: HASUG demo

Metadata Identifier: A0000001.WHDWENV.A0000001

Figure 15

The resulting HTML generates a header and table of contents similar to what is shown in Figure 16.



Process Documentation

HASUG demo

Generated: 12JUL2002:20:54:49

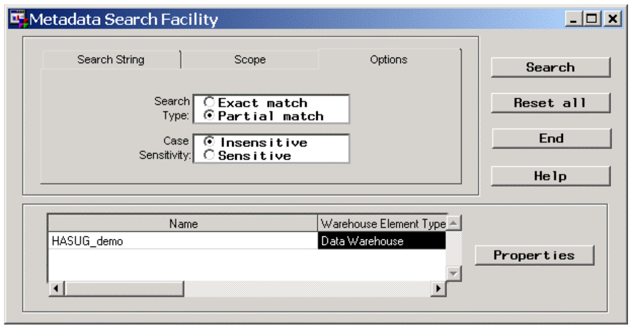
Table of Contents

- Description
- Notes
- Columns
- Input Sources
- Dependencies
- Source Code

Figure 16

SEARCHING AND MIGRATING METADATA

One of the big advantages of entering all of the warehouse details as metadata is that one can search it. From the pull-down menu, select Tools -> Search Metadata… The dialog box similar to Figure 17 should appear.



Search Type: Exact match [✓ Partial match]
Case Sensitivity: [✓ Insensitive] Sensitive

Name	Warehouse Element Type
HASUG_demo	Data Warehouse

Figure 17

Enter the search string. Because the search is conducted on metadata, it can be restricted by warehouse element type. To go directly to an item shown in the results window, just double-click on the Warehouse Element Type.

Another useful tool is the Metadata Copy wizard. If one needs to move metadata to a different directory path, go to the SAS/Warehouse Administrator desktop. Right-click on the warehouse environment to be copied and select Copy… Follow the instructions given by the wizard.

One of the author's tricks when modeling warehouses on his laptop is to map a project to the same drive letter and path as is used on the client's host computer. To migrate the warehouse to host computer, he merely copies the metadata physical directory to a CD-ROM or Zip disk and then copies it to the host computer.

FUTURE CHANGES AND ENHANCEMENTS

At SUGI 27, the author visited with some of SAS staff responsible for future versions of SAS/Warehouse Administrator. Among some of the improvements anticipated for future releases were:

- Multiple-table join tools
- Enhancements to take advantage of multi-threading in SAS Version 9
- Integration of the File Import Wizard

Also on the horizon was a new version of the product called Data Builder. Data Builder provides a Java interface so it will no longer be necessary to be sitting in front of the host computer or to operate it via terminal emulation software. The Java interface will communicate to a metadata repository and server.

There will be a one-way conversion tool to migrate SAS/Warehouse Administrator metadata into Data Builder. However, existing SAS/Warehouse Administrator users can continue to use the product as in the past.

CONCLUSION

The author hopes this paper has explained his passion for using SAS/Warehouse Administrator over traditional methods for creating and maintaining data warehouses and marts. He also hopes that this paper clearly illustrated how data warehouses are modeled in SAS/Warehouse Administrator and highlighted how metadata is created and managed. Last, the author hopes that the tips passed by this paper will reduce the learning curve by other users of this product.

ACKNOWLEDGEMENTS

AppDev Studio, SAS, SAS/ACCESS, SAS/SHARE, SAS/Warehouse Administrator, and WebHound are trademarks of SAS Institute Inc. Microsoft, SQL Server, and Microsoft Windows are trademarks of the Microsoft Corporation. Oracle is a registered trademark of Oracle Corporation.

The author would like to thank the Hartford Area SAS User Group Steering Committee, which encouraged him to prepare this paper. Special thanks also go to Jon Schiltz and Tina Hobbs, SAS Technical Support Department, and to the author's colleagues at The Nash Engineering Company and Pfizer Inc.

CONTACT INFORMATION

The author may be contacted as follows:

Michael L. Davis
Bassett Consulting Services, Inc.
10 Pleasant Drive
North Haven CT 06473-3712
E-Mail: michael@bassettconsulting.com
Web: http://www.bassettconsulting.com
Telephone: (203) 562-0640
Facsimile: (203) 498-1414