THE POWER TO KNOW®

SAS® Data Integration Studio 3.4

User's Guide

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2007. *SAS® Data Integration Studio 3.4: User's Guide*. Cary, NC: SAS Institute Inc.

SAS® Data Integration Studio 3.4: User's Guide

Copyright © 2002–2007, SAS Institute Inc., Cary, NC, USA

ISBN 978-1-59994-198-1

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, May 2007

Contents

PART 1

Introduction

CHAPTER 1 Introduction to SAS Data Integration

About SAS Data Integration

Data integration is the process of consolidating data from a variety of sources in order to produce a unified view of the data. SAS supports data integration in the following ways:

- *Connectivity and metadata*. A shared metadata environment provides consistent data definition across all data sources. SAS software enables you to connect to, acquire, store, and write data back to a variety of data stores, streams, applications, and systems on a variety of platforms and in many different environments. For example, you can manage information in Enterprise Resource

Planning (ERP) systems; relational database management systems (RDBMS), flat files, legacy systems, message queues, and XML.

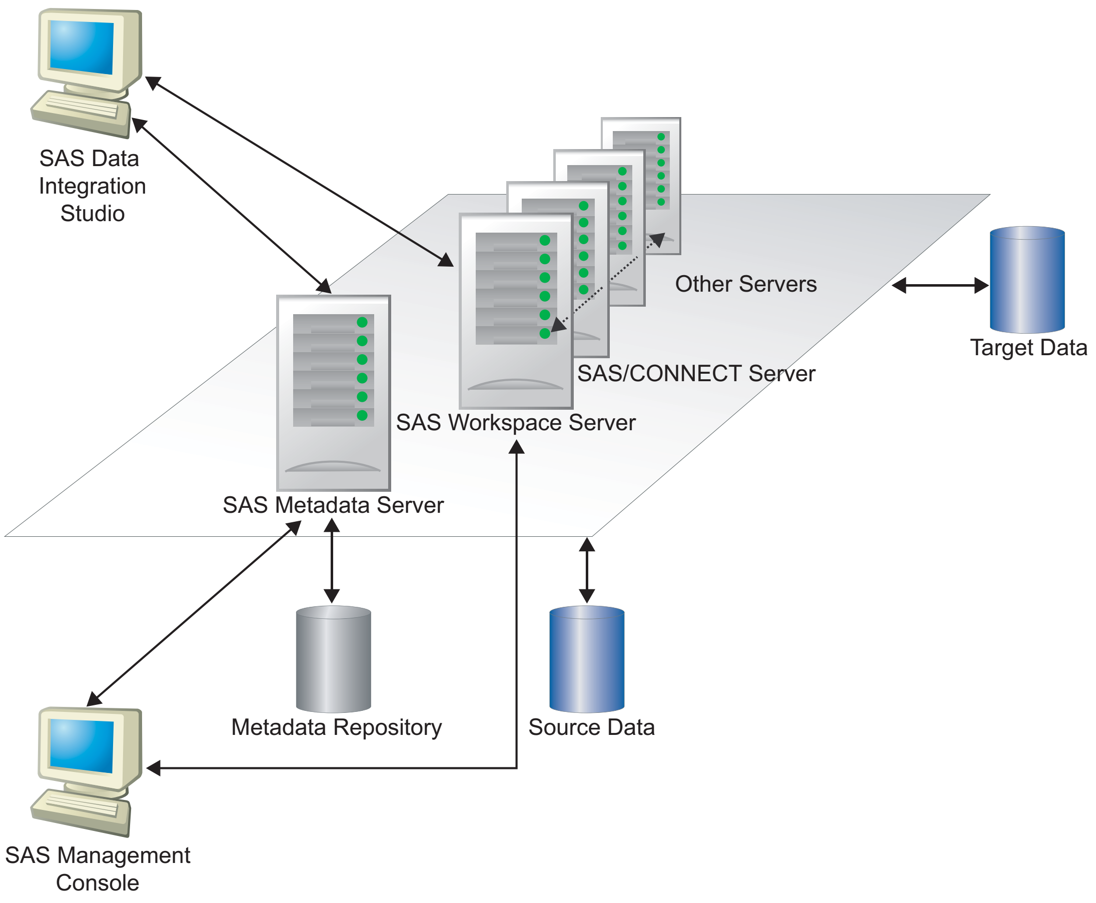
- *Data cleansing and enrichment*. Integrated SAS Data Quality software enables you to profile, cleanse, augment, and monitor data to create consistent, reliable information. SAS Data Integration Studio provides a number of transformations and functions that can improve the quality of your data.
- *Extraction, transformation, and loading (ETL)*. SAS Data Integration Studio enables you to extract, transform, and load data from across the enterprise to create consistent, accurate information. It provides a point-and-click interface that enables designers to build process flows, quickly identify inputs and outputs, and create business rules in metadata, all of which enable the rapid generation of data warehouses, data marts, and data streams.
- *Migration and synchronization*. SAS Data Integration Studio enables you to migrate, synchronize, and replicate data among different operational systems and data sources. Data transformations are available for altering, reformatting, and consolidating information. Real-time data quality integration allows data to be cleansed as it is being moved, replicated, or synchronized, and you can easily build a library of reusable business rules.
- *Data federation*. SAS Data Integration Studio enables you to query and use data across multiple systems without the physical movement of source data. It provides virtual access to database structures, ERP applications, legacy files, text, XML, message queues, and a host of other sources. It enables you to join data across these virtual data sources for real-time access and analysis. The semantic business metadata layer shields business staff from underlying data complexity.
- *Master data management*. SAS Data Integration Studio enables you to create a unified view of enterprise data from multiple sources. Semantic data descriptions of input and output data sources uniquely identify each instance of a business element (such as customer, product, and account) and standardize the master data model to provide a single source of truth. Transformations and embedded data quality processes ensure that master data is correct.

A Basic Data Integration Environment

Overview of a Data Integration Environment

The following figure shows the main clients and servers in a SAS data integration environment.

Figure 1.1 SAS Data Integration Studio Environment



Administrators use SAS Management Console to connect to a SAS Metadata Server. They enter metadata about servers, libraries, and other resources on your network and save this metadata to a repository. SAS Data Integration Studio users connect to the same metadata server and register any additional libraries and tables that they need. Then, they create process flows that read source tables and create target tables in physical storage.

SAS Management Console

SAS Management Console provides a single interface through which administrators can explore and manage metadata repositories. With this interface, administrators can efficiently set up system resources, manage user and group accounts, and administer security.

SAS Data Integration Studio

SAS Data Integration Studio is a visual design tool that enables you to consolidate and manage enterprise data from a variety of source systems, applications, and technologies. This software enables you to create process flows that accomplish the following tasks:

- extract, transform, and load data for use in data warehouses and data marts

- cleanse, migrate, synchronize, replicate, and promote data for applications and business services

SAS Data Integration Studio enables you to create metadata that defines sources, targets, and the processes that connect them. This metadata is stored in one or more shareable repositories. SAS Data Integration Studio uses the metadata to generate or retrieve SAS code that reads sources and creates targets in physical storage. Other applications that share the same repositories can use the metadata to access the targets and use them as the basis for reports, queries, or analyses.

Through its metadata, SAS Data Integration Studio provides a single point of control for managing the following resources:

- data sources (from any platform that is accessible to SAS and from any format that is accessible to SAS)
- data targets (to any platform that is accessible to SAS, and to any format that is supported by SAS)
- processes that specify how data is extracted, transformed, and loaded from a source to a target
- jobs that organize a set of sources, targets, and processes (transformations)
- source code generated by SAS Data Integration Studio
- user-written source code

Note: SAS Data Integration Studio was formerly named SAS ETL Studio. △

Servers

SAS Application Servers

When the SAS Intelligence Platform was installed at your site, a metadata object that represents the SAS server tier in your environment was defined. In the SAS Management Console interface, this type of object is called a SAS Application Server. If you have a SAS server, such as a SAS Workspace Server, on the same machine as your SAS Metadata Server, the application server object is named **`SASMain`**; otherwise, it is named **`SASApp`**.

A SAS Application Server is not an actual server that can execute SAS code submitted by clients. Rather, it is a logical container for a set of application server components, which do execute code—typically SAS code, although some components can execute Java code or MDX queries. For example, a SAS Application Server might contain a workspace server, which can execute SAS code that is generated by clients such as SAS Data Integration Studio. A SAS Application Server might also contain a stored process server, which executes SAS Stored Processes, and a SAS/CONNECT Server, which can upload or download data and execute SAS code submitted from a remote machine.

The following table lists the main SAS Application Server components and describes how each one is used.

Table 1.1 SAS Application Servers

Server	How Used	How Specified
SAS Metadata Server	Reads and writes metadata in a SAS Metadata Repository.	In each user's metadata profile.
SAS Workspace Server	Executes SAS code; reads and writes data.	As a component in a SAS Application Server object.
SAS/ CONNECT Server	Submits generated SAS code to machines that are remote from the default SAS Application Server; can also be used for interactive access to remote libraries.	As a component in a SAS Application Server object.
SAS OLAP Server	Creates cubes and processes queries against cubes.	As a component in a SAS Application Server object.
Stored Process Server	Submits stored processes for execution by a SAS session. Stored processes are SAS programs that are stored and can be executed by client applications.	As a component in a SAS Application Server object.
SAS Grid Server	Supports a compute grid that can execute grid-enabled jobs created in SAS Data Integration Studio.	As a component in a SAS Application Server object.

Typically, administrators install, start, and register SAS Application Server components. SAS Data Integration Studio users are told which SAS Application Server object to use.

SAS Data Servers

The following table lists two special-purpose servers for managing SAS data.

Table 1.2 SAS Data Servers

Server	How Used	How Specified
SAS/SHARE Server	Enables concurrent access of server libraries from multiple users.	In a SAS/SHARE library.
SAS Scalable Performance Data (SPD) Server	Provides parallel processing for large SAS data stores; provides a comprehensive security infrastructure, backup and restore utilities, and sophisticated administrative and tuning options.	In an SPD Server library.

Typically, administrators install, start, and register these servers and register the SAS/SHARE library or the SPD Server library. SAS Data Integration Studio users are told which library to use.

Database Management System (DBMS) Servers

SAS Data Integration Studio uses a SAS Application Server and a database server to access tables in database management systems such as Oracle and DB2.

When you start a source designer or a target designer, the wizard tries to connect to a SAS Application Server. You are then prompted to select an appropriate database library. SAS Data Integration Studio uses the metadata for the database library to generate a SAS/ACCESS LIBNAME statement, and the statement is submitted to the SAS Application Server for execution.

The SAS/ACCESS LIBNAME statement specifies options that are required to communicate with the relevant database server. The options are specific to the DBMS to which you are connecting. For example, here is a SAS/ACCESS LIBNAME statement that could be used to access an Oracle database:

```
libname mydb oracle user=admin1 pass=ad1min path='V2o7223.world'
```

Typically, administrators install, start, and register DBMS servers and register the DBMS libraries. SAS Data Integration Studio users are told which library to use.

Enterprise Resource Management Servers

Optional data surveyor wizards can be installed that provide access to the metadata and data from enterprise applications. Applications from vendors such as SAP, Oracle, PeopleSoft, and Siebel are supported. Typically, administrators install, start, and register ERP servers. SAS Data Integration Studio users are told which server metadata to use.

Libraries

In SAS software, a library is a collection of one or more files that are recognized by SAS and that are referenced and stored as a unit. Libraries are critical to SAS Data Integration Studio. You cannot begin to enter metadata for sources, targets, or jobs until the appropriate libraries have been registered in a metadata repository.

Accordingly, one of the first tasks in a SAS Data Integration Studio project is to specify metadata for the libraries that contain sources, targets, or other resources. At some sites, an administrator adds and maintains most of the libraries that are needed, and the administrator tells SAS Data Integration Studio users which libraries to use. The steps for specifying metadata about a Base SAS library are described in "Registering Any Libraries That You Need" on page 55.

Additional Information

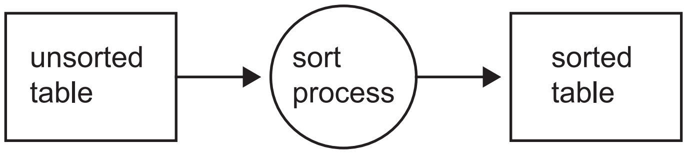
For more information about setting up a data integration environment, administrators should see "Administrative Documentation for SAS Data Integration Studio" on page 12.

Overview of Building a Process Flow

Problem

You want to become familiar with SAS Data Integration Studio, so you decide to create a simple process flow that reads data from a source table, sorts the data, and then writes the sorted data to a target table, as shown in the following figure.

Figure 1.2 Simple Process Flow



Solution

Create a job in SAS Data Integration Studio that specifies the desired process flow. Perform the following tasks:

- □ Connect to a metadata server.
- □ Register the source table.
- □ Register the target table.
- □ Create an empty job.
- □ Drag and drop the SAS Sort transformation on the job.
- □ Drag and drop the source table metadata and target table metadata on the job.
- □ Update the metadata for the tables and the SAS Sort transformation as needed for your environment.
- □ Execute the job.

It is assumed that administrators have installed, configured, and registered the relevant servers, libraries, and other resources that are required to support SAS Data Integration Studio in your environment.

Tasks

Connect to the Metadata Server

Most servers, data, and other resources on your network are not available to SAS Data Integration Studio until they are registered in a repository on a SAS Metadata Server. Accordingly, when you start SAS Data Integration Studio, you are prompted to select a metadata profile which specifies a connection to a metadata server. You might have a number of different profiles that connect to different metadata servers at your site. Select the profile that will connect to the metadata server with the metadata that you will need during the current session.

For details about creating a metadata profile, see "Connecting to a Metadata Server" on page 51.

Register Source Tables

Suppose that the source table in the example process flow is an existing SAS table, but the table is not currently registered; that is, metadata about this table has not been saved to the current metadata server. One way to register a table that exists in physical storage is to use a source designer wizard. To display the source designer wizard for a SAS table, select **Tools ▶ Source Designer** from the menu bar. A selection window displays. Click **SAS**, and then click **OK**. The SAS source designer displays. A source designer wizard enables you to:

- □ specify the library that contains the table to be registered (typically, this library has been registered ahead of time)
- □ display a list of tables contained in the selected library
- □ select one or more tables in that library
- □ generate and save metadata for the selected tables

For details about using source designers, see "Registering Tables with a Source Designer" on page 85.

Register Target Tables

Suppose that the target table in the example process flow is a new table, one that does not yet exist in physical storage. You could use the Target Table wizard to specify metadata for the table. Later, you can drag and drop this metadata on the target position in a process flow. When the process flow is executed, SAS Data Integration Studio will use the metadata for the target table to create a physical instance of that table.

One way to register a table that does not exist in physical storage is to use the Target Table wizard. To display the Target Table wizard, select **Tools ▶ Target Designer** from the menu bar. A selection window displays. Click **Target Table**, and then click **OK**. The Target Table wizard displays.

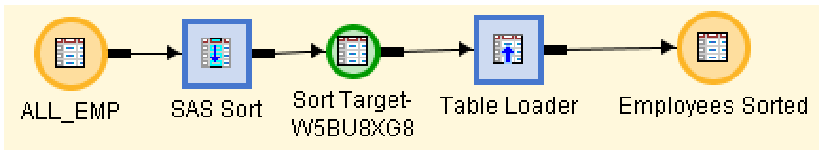
The Target Table wizard enables you to specify the physical location, column structure, and other attributes of the target table and save that metadata to the current repository.

For details about using the Target Table wizard, see "Registering Tables with the Target Table Wizard" on page 87.

Create a Job That Specifies the Desired Process Flow

In SAS Data Integration Studio, a process flow is contained in a job. One way to create a job is to use the New Job wizard to create an empty job, then drag and drop metadata for the source tables, the target tables, and the desired transformations onto the empty job, and build the desired process flow. For details about this method, see "Creating an Empty Job" on page 143. For now, assume that you have used this method to create the process flow shown in the following display.

Display 1.1 Process Flow Diagram for a Job That Sorts Data



Given the direction of the arrows in the previous display:

- ALL_EMP specifies metadata for the source table.
- SAS Sort specifies metadata for the sort process, which writes its output to a temporary output table, Sort Target-W5BU8XGB. (For more information about temporary output tables, see "Manage Temporary and Permanent Tables for Transformations" on page 239.)
- Table Loader specifies metadata for a process that reads the output from the previous step and loads this data into a target table.
- Employees Sorted specifies metadata for the target table.

SAS Data Integration Studio uses the preceding process flow diagram to generate SAS code that reads ALL_EMP, sorts this information, and writes the sorted information to a temporary output table. Then, the information is written to the Employees Sorted table.

Run the Job

One way to execute a SAS Data Integration Studio job is to select **Process ► Submit** from the menu bar. The code is then submitted to a SAS Application Server, which executes the code. If the job is successful, the output of the job is created or updated.

Next Tasks

The output from a job can become the source for another job in SAS Data Integration Studio, or it can be the source for a report or query in another SAS application. Any tables, libraries, or other resources that were registered in order to create the job are also available to other SAS applications that connected to the same metadata repository.

Impact of Change Management

The change management feature adds a few steps to some of the previous tasks. For more information, see "Working with Change Management" on page 59.

Advantages of SAS Data Integration

SAS data integration projects have a number of advantages over projects that rely heavily on custom code and multiple tools that are not well integrated.

- SAS data integration reduces development time by enabling the rapid generation of data warehouses, data marts, and data streams.
- It controls the costs of data integration by supporting collaboration, code reuse, and common metadata.
- It increases returns on existing IT investments by providing multi-platform scalability and interoperability.
- It creates process flows that are reusable, easily modified, and have embedded data quality processing. The flows are self-documenting and support data lineage analysis.

Online Help for SAS Data Integration Studio

The online Help describes all windows in SAS Data Integration Studio, and it summarizes the main tasks that you can perform with the software. The Help includes examples for all source designer wizards, all target designer wizards, and all transformations in the Process Library. The Help also includes a What's New topic and a set of Usage Note topics for the current version of the software.

Perform the following steps to display the main Help window for SAS Data Integration Studio.

1 Start SAS Data Integration Studio as described in "Starting SAS Data Integration Studio" on page 50.

2 From the menu bar, select **Help ▶ Contents**. The main Help window displays.

To display the Help for an active window or tab, click its **`Help`** button. If the window or tab does not have a **`Help`** button, press the **`F1`** key.

To search for topics about concepts or features that are identified by specific words, such as "application server," display the main Help window. Then, click the **`Search`** tab (magnifying glass icon). Enter the text to be found and press the **`Enter`** key.

Administrative Documentation for SAS Data Integration Studio

Many administrative tasks, such as setting up the servers that are used to execute jobs, are performed outside of the SAS Data Integration Studio interface. Such tasks are described in SAS Intelligence Platform documentation, which can be found at the following location: **`http://support.sas.com/913administration`**.

The following table identifies the main SAS Intelligence Platform documentation for SAS Data Integration Studio.

Table 1.3 SAS Intelligence Platform Documentation for SAS Data Integration Studio

Administrative Task	Related Documentation
□ Set up metadata servers and metadata repositories.	*SAS Intelligence Platform: System Administration Guide*
□ Set up data servers and libraries for common data sources.	*SAS Intelligence Platform: Data Administration Guide*

Administrative Task	Related Documentation
□ Set up SAS Application Servers. □ Set up grid computing (so that jobs can execute on a grid).	*SAS Intelligence Platform: Application Server Administration Guide*
□ Set up change management. □ Manage operating system privileges on target tables (job outputs). □ Set up servers and libraries for remote data (multi-tier environments). □ Set up security for Custom tree folders. □ Set up a central repository for importing and exporting generated transformations. □ Set up support for message queue jobs. □ Set up support for Web service jobs and other stored process jobs. □ Enable the bulk-loading of data into target tables in a DBMS. □ Set up SAS Data Quality software. □ Set up support for job status handling. □ Set up support for FTP and HTTP access to external files.	*SAS Intelligence Platform: Desktop Application Administration Guide*

Accessibility Features in SAS Data Integration Studio

Accessibility Standards

SAS Data Integration Studio includes features that improve usability of the product for users with disabilities. These features are related to accessibility standards for electronic information technology that were adopted by the U.S. Government under Section 508 of the U.S. Rehabilitation Act of 1973, as amended. SAS Data Integration Studio supports Section 508 standards except as noted in the following table.

Table 1.4 Accessibility Exceptions

Section 508 Accessibility Criteria	**Support Status**	**Explanation**
(a) When software is designed to run on a system that has a keyboard, product functions shall be executable from a keyboard where the function itself or the result of performing a function can be discerned textually.	Supported with exceptions	The software supports keyboard equivalents for all user actions. Tree controls in the user interface can be individually managed and navigated through using the keyboard. However, some exceptions exist. Some ALT key shortcuts are not functional. Also, some more advanced manipulations require a mouse. Still, the basic functionality for displaying trees in the product is accessible from the keyboard. Based on guidance from the Access Board, keyboard access to drawing tasks does not appear to be required for compliance with Section 508 standards. Accordingly, keyboard access does not appear to be required for the **`Process Editor`** tab in the Process Designer window, or the **`Designer`** tab in the SQL Join properties window. Specifically, use of the **`Process Editor`** tab in the Process Flow Diagram and the **`Designer`** tab in the SQL Join Properties window are functions that cannot be discerned textually. Both involve choosing a drawing piece, dragging it into the workspace, and designing a flow. These tasks required a level of control that is provided by a pointing device. Moreover, the same result can be achieved by editing the source code for flows. **`Example:`** Use of the **`Process Editor`** tab in the Process Flow Diagram is designed for visual rather than textual manipulation. Therefore, it cannot be operated via keyboard. If you have difficulty using a mouse, then you can create process flows with user-written source code. See Chapter 12, "Working with User-Written Code," on page 215.
(c) A well-defined on-screen indication of the current focus shall be provided that moves among interactive interface elements as the input focus changes. The focus shall be programmatically exposed so that Assistive Technology can track focus and focus changes.	Supported with exceptions	In some wizards, when focus is on an element in a wizard pane, rather than a button, focus is not apparent. If an element in the pane is highlighted and focus is moved to a button, the element appearance is unchanged, so the user might not be certain when focus is on such an item. **`Example:`** When you launch the Target Designer and press the down arrow, you can traverse the Targets tree to select the type of target that you want to design even though no object has visual focus.

Section 508 Accessibility Criteria	**Support Status**	**Explanation**
(d) Sufficient information about a user interface element including the identity, operation, and state of the element shall be available to Assistive Technology. When an image represents a program element, the information conveyed by the image must also be available in text.	Supported with exceptions	In some wizards, identity, operation, and state of some interface elements is ambiguous. SAS currently plans to address this in a future release. **`Example:`** When you select a library in the Source Designer wizard, you must use the SAS Library combo box. If you are using the JAWS screen reader, the reader immediately reads not only the library name but also all of its details. If you want to know the libref, you must know that the label exists and that its shortcut is Alt+F. Then, you must press Alt+F so that the JAWS screen reader will read the label and its read-only text. You can move among the items in Library Details only after you use a shortcut to get to one of them.
(g) Applications shall not override user selected contrast and color selections and other individual display attributes.	Supported with exceptions	When the user sets the operating system settings to high contrast, some attributes of that setting are not inherited. **`Example:`** As with most other Java applications, system font settings are not inherited in the main application window. If you need larger fonts, consider using a screen magnifier.
(l) When electronic forms are used, the form shall allow people using Assistive Technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.	Supported with exceptions	When navigating with a keyboard to choose a path in the Browse dialog box, the focus disappears. To work around the problem, either (1) count the number of times you press the TAB key and listen closely to the items, or (2) type the path explicitly. **`Example:`** In some wizards such as the Source Designer, the visual focus can disappear sometimes when you operate the software with only a keyboard. If so, continue to press the TAB key until an interface element regains focus.

If you have questions or concerns about the accessibility of SAS products, send e-mail to **`accessibility@sas.com`**.

Enabling Assistive Technologies

For instructions on how to configure SAS Data Integration Studio software so that assistive technologies will work with the application, see the information about downloading the Java Access Bridge in the section about accessibility features in the *SAS Intelligence Platform: Desktop Application Administration Guide*.

CHAPTER
2

About the Main Windows and Wizards

Overview of the Main Windows and Wizards

The windows and wizards in SAS Data Integration Studio can be divided into categories that facilitate working with metadata, jobs, and the SAS Data Integration Studio application as a whole. The wizards are available from the shortcut bar or from the tools menu on the SAS Data Integration Studio desktop. The following table lists the metadata windows and wizards.

Table 2.1 Metadata Windows

Window or wizard	**Enables you to**
Property windows for tables and other objects	View and update the metadata for tables or external files, jobs, and transformations.
Windows for viewing data	Display the data in tables or external files. Includes the View Data, View File, and View Statistics windows.
Impact analysis windows	Identify the tables, columns, jobs, and transformations that are affected by a change in a selected table or column; identify the tables, columns, jobs, and transformations that contribute to the content of a selected table or column.
Import/Export Metadata Wizards	Export metadata from and import metadata into SAS Data Integration Studio.
Source Designer Wizards	Register one or more tables that exist in physical storage.
Target Table wizard	Register a new table that will be created when a SAS Data Integration Studio job is executed; register a new table that reuses column metadata from one or more registered tables.

Window or wizard	Enables you to
Cube wizards	Create or maintain SAS cubes.
Data surveyors	Extract, search, and navigate data from SAP, Siebel, PeopleSoft, Oracle, and other enterprise application vendors.

For more information about these components, see “Metadata Windows and Wizards” on page 20. The following table lists the job windows and wizards.

Table 2.2 Job Windows

Window or wizard	Enables you to
Process Designer window	Create process flows; generate and submit code for jobs.
Expression Builder	Create SAS expressions that aggregate columns, perform conditional processing, and perform other tasks in SAS Data Integration Studio jobs.
Job Status Manager window	View the status of jobs that have been submitted for execution. You can also cancel, kill, or resubmit jobs.
Source Editor window	Write SAS code and submit it for execution.
New Job wizard	Select one or more tables as the targets (outputs) of a job Alternatively, you can create an empty job onto which you can drag and drop transformations and tables.
Transformation Generator wizard	Specify user-written SAS code for a generated transformation and save metadata for the transformation to the current repository.

For more information about these components, see “Job Windows and Wizards” on page 27. The following table lists the application windows.

Table 2.3 SAS Data Integration Application Windows

Window	Enables you to
Metadata Profile window	Select or create a metadata profile. You can also connect to the metadata server that is specified in the profile.
Desktop window	Work with the menus, trees, windows, and other objects that comprise the user interface for SAS Data Integration Studio.

Window	**Enables you to**
Options window	Specify global options for SAS Data Integration Studio.
Tree views	Display and work with metadata in the repositories that are specified in the current metadata profile. The following trees are possible, depending on configuration: □ Inventory □ Custom □ Process Library □ Project □ Quick Properties □ Comparison Results □ Metadata

For more information about these components, see "SAS Data Integration Studio Application Windows" on page 32.

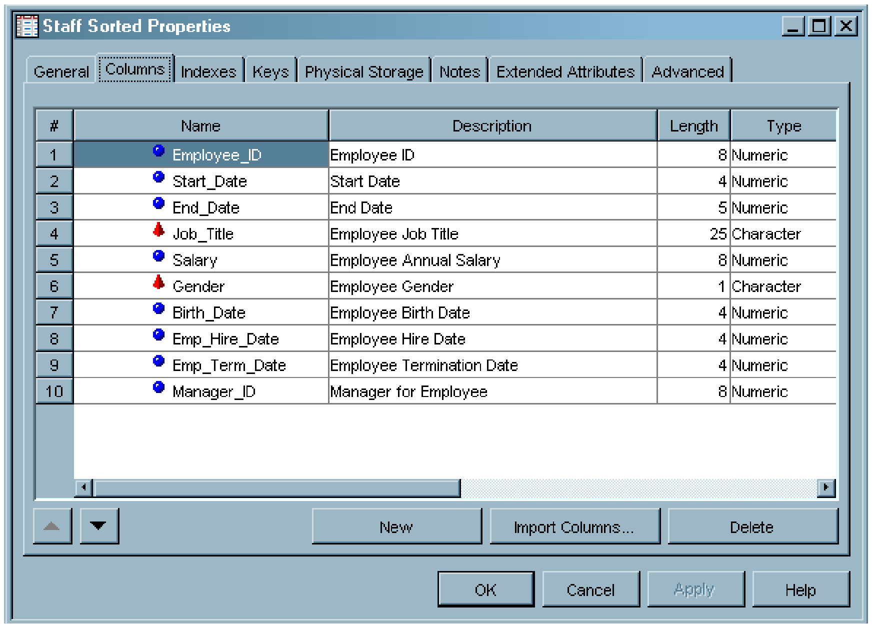
Metadata Windows and Wizards

Property Windows for Tables and Other Objects

Property Window for a Table or External File

Use the properties window for a table or an external file to view or update the metadata for its columns and other attributes. The following display shows a typical window.

Display 2.1 Table Properties Window



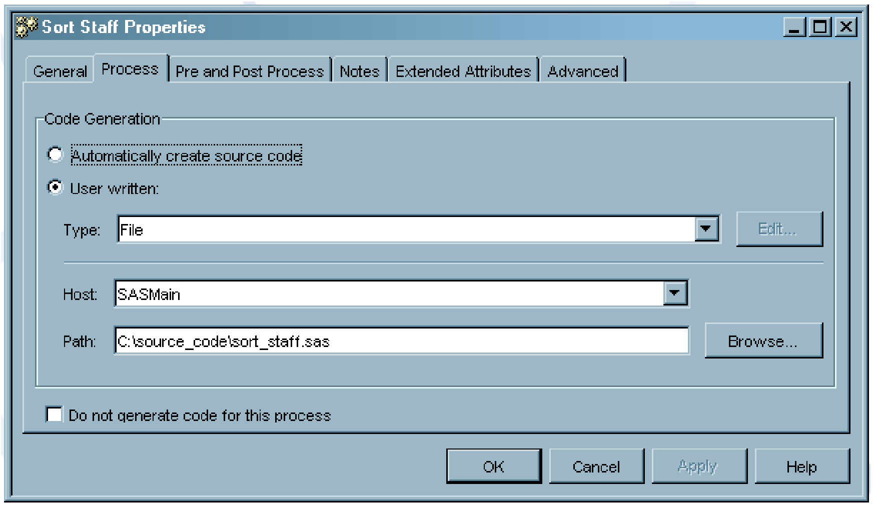
#	Name	Description	Length	Type
1	Employee_ID	Employee ID	8	Numeric
2	Start_Date	Start Date	4	Numeric
3	End_Date	End Date	5	Numeric
4	Job_Title	Employee Job Title	25	Character
5	Salary	Employee Annual Salary	8	Numeric
6	Gender	Employee Gender	1	Character
7	Birth_Date	Employee Birth Date	4	Numeric
8	Emp_Hire_Date	Employee Hire Date	4	Numeric
9	Emp_Term_Date	Employee Termination Date	4	Numeric
10	Manager_ID	Manager for Employee	8	Numeric

The window shown in the previous display contains the metadata for a typical table. For a summary of how to use this window, see "Viewing or Updating Table Metadata" on page 89.

Property Window for a Job

Use the properties window for a job to view or update its basic metadata. For example, you can specify whether the code for the current job will be generated by SAS Data Integration Studio or will be retrieved from a specified location. You can also use this window to specify code that should be run before or after a job executes. The following display shows a typical window.

Display 2.2 Job Properties Window



Code Generation

( ) Automatically create source code

(●) User written:

Type: File

Host: SASMain

Path: C:\source_code\sort_staff.sas

☐ Do not generate code for this process

If you want to specify user-written code for the job, you can enter metadata that is similar to the metadata that is shown in the previous display. In the job properties window shown in the previous display, the **`User written`** option has been selected, and the physical path to a source code file has been specified.

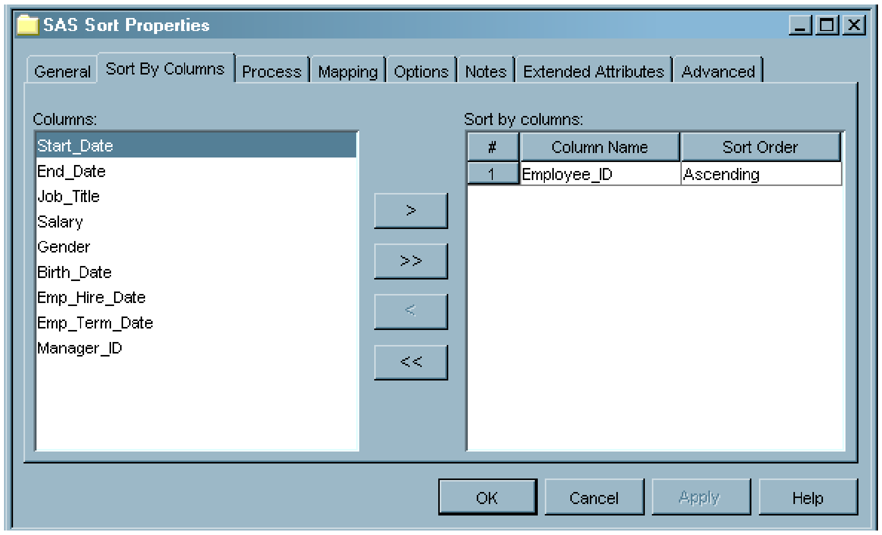
If you wanted to execute code before or after the Sort Staff job is executed, you can click the **`Pre and Post Process`** tab and specify the code. For example, you might want to issue a SAS LIBNAME statement before the job is run.

For a summary of how to use the job properties window, see "Viewing or Updating Job Metadata" on page 150.

Transformation Property Window

Use a transformation properties window to view or update the metadata for a process in a job. The metadata for a transformation specifies how SAS Data Integration Studio will generate code for the corresponding process. The window for each kind of transformation has one or more tabs that are unique to the corresponding process. The following display shows a typical window.

Display 2.3 Transformation Properties Window



Columns:

Start_Date
End_Date
Job_Title
Salary
Gender
Birth_Date
Emp_Hire_Date
Emp_Term_Date
Manager_ID

Sort by columns:

#	Column Name	Sort Order
1	Employee_ID	Ascending

The window shown in the previous display contains the metadata for the SAS Sort transformation, which is described in Chapter 16, "Working with SAS Sorts," on page 275. Note that the rows in the output table for this transformation will be sorted by employee ID. For a summary of how to use transformation property windows, see "Viewing or Updating the Metadata for Transformations" on page 197.

Windows Used to View Data

View Data Window

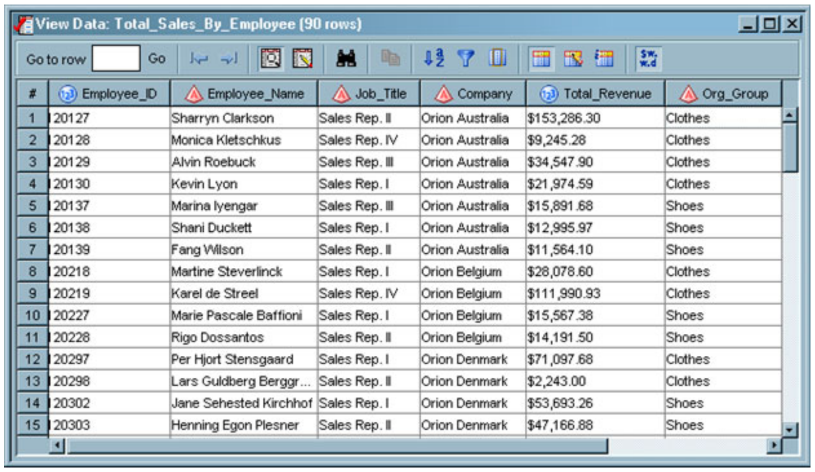
The View Data window is available in the tree views on the desktop. It works in two modes, browse and edit. The browse mode enables you to view the data displayed in a SAS table or view, in an external file, in a temporary output table displayed in a process flow diagram, or in a DBMS table or view that is part of a SAS library for DBMS data stores. The table, view, or external file must be registered in a current metadata repository and must exist in physical storage. However, temporary output tables are retained until the Process Designer window is closed or the current server session is ended in some other way (for example, by selecting **Process** ▶ **Kill** from the menu bar).

Use the edit mode to perform simple editing operations on the data in the View Data window. For example, you can overwrite the data in a cell, copy and paste rows of data, and delete data. You can even create completely new tables. However, this editing mode is enabled only on SAS tables that are stored in a BASE engine library and are assigned on a workspace server.

The View Data window typically uses the metadata for a data store to format the data for display. Accordingly, the View Data window can be used to verify that the metadata for a data store is appropriate for use in the intended job. If the window does not correctly display the data in the selected data store, then you might have to update the corresponding metadata before you use it in a job.

The following display shows a typical View Data window.

Display 2.4 View Data Window



#	Employee_ID	Employee_Name	Job_Title	Company	Total_Revenue	Org_Group
1	20127	Sharryn Clarkson	Sales Rep. II	Orion Australia	$153,286.30	Clothes
2	20128	Monica Kletschkus	Sales Rep. IV	Orion Australia	$9,245.28	Clothes
3	20129	Alvin Roebuck	Sales Rep. III	Orion Australia	$34,547.90	Clothes
4	20130	Kevin Lyon	Sales Rep. I	Orion Australia	$21,974.59	Clothes
5	20137	Marina Iyengar	Sales Rep. III	Orion Australia	$15,891.68	Shoes
6	20138	Shani Duckett	Sales Rep. I	Orion Australia	$12,995.97	Shoes
7	20139	Fang Wilson	Sales Rep. II	Orion Australia	$11,564.10	Shoes
8	20218	Martine Steverlinck	Sales Rep. I	Orion Belgium	$28,078.60	Clothes
9	20219	Karel de Streel	Sales Rep. IV	Orion Belgium	$111,990.93	Clothes
10	20227	Marie Pascale Baffioni	Sales Rep. I	Orion Belgium	$15,567.38	Shoes
11	20228	Rigo Dossantos	Sales Rep. II	Orion Belgium	$14,191.50	Shoes
12	20297	Per Hjort Stensgaard	Sales Rep. I	Orion Denmark	$71,097.68	Clothes
13	20298	Lars Guldberg Berggr...	Sales Rep. II	Orion Denmark	$2,243.00	Clothes
14	20302	Jane Sehested Kirchhof	Sales Rep. I	Orion Denmark	$53,693.26	Shoes
15	20303	Henning Egon Plesner	Sales Rep. II	Orion Denmark	$47,166.88	Shoes

The title bar in the View Data window displays the name of the object that is being viewed and the total number of rows. If a column has a description, the description displays in the column heading in the View Data window. Otherwise, the physical name of the column displays in the column heading. A round icon to the left of the name indicates that the column is numeric, and a pyramid-shaped icon to the left of the name indicates that the column contains character data.

To customize the data view displayed in the View Data window, right-click on a column name, row number, or table cell. Then, select an appropriate option from the pop-up menu. To display Help for the View Data window, press F1. See also "Browsing Table Data" on page 109.

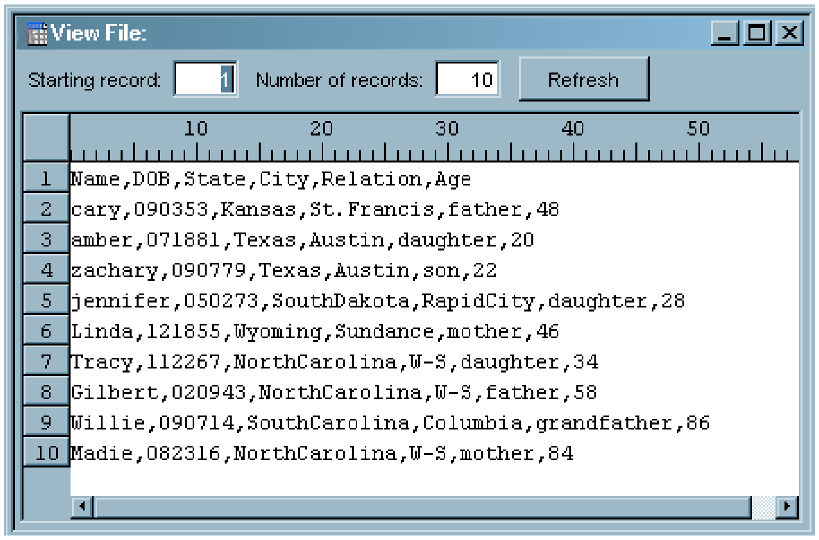
View File Window

Use the View File window to display the raw contents of an external file. Unlike the View Data window, the View File window does not use SAS metadata to format the contents of the corresponding external file. The View File option reads the structure of the external file directly and displays the data accordingly.

The external file must exist in physical storage. You cannot use the View File window to view an external file that is accessed with user-written code.

For an example of the View File window, please see the following display.

Display 2.5 View File Window

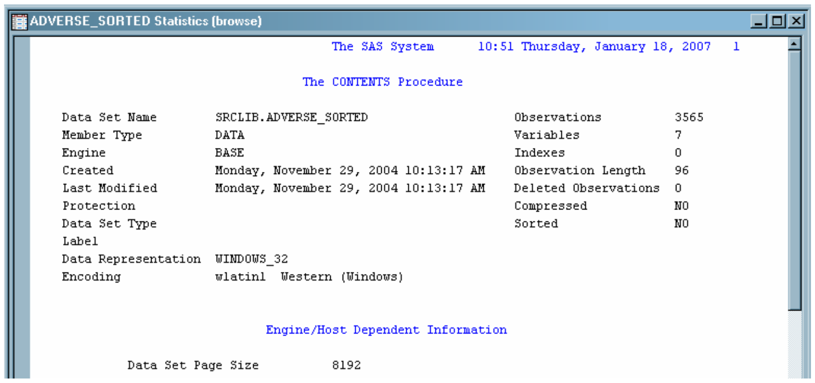


```
Name,DOB,State,City,Relation,Age
cary,090353,Kansas,St.Francis,father,48
amber,071881,Texas,Austin,daughter,20
zachary,090779,Texas,Austin,son,22
jennifer,050273,SouthDakota,RapidCity,daughter,28
Linda,121855,Wyoming,Sundance,mother,46
Tracy,112267,NorthCarolina,W-S,daughter,34
Gilbert,020943,NorthCarolina,W-S,father,58
Willie,090714,SouthCarolina,Columbia,grandfather,86
Madie,082316,NorthCarolina,W-S,mother,84
```

View Statistics Window

Use the View Statistics window to see or print statistics for a table. Table statistics might include size, number of rows, number of populated rows, constraints, and other information that is generally available through SAS. An example is shown in the following display:

Display 2.6 View Statistics Window



```
                                The SAS System      10:51 Thursday, January 18, 2007   1

                              The CONTENTS Procedure

Data Set Name        SRCLIB.ADVERSE_SORTED                   Observations          3565
Member Type          DATA                                    Variables             7
Engine               BASE                                    Indexes               0
Created              Monday, November 29, 2004 10:13:17 AM   Observation Length    96
Last Modified        Monday, November 29, 2004 10:13:17 AM   Deleted Observations  0
Protection                                                   Compressed            NO
Data Set Type                                                Sorted                NO
Label
Data Representation  WINDOWS_32
Encoding             wlatin1  Western (Windows)


                         Engine/Host Dependent Information


         Data Set Page Size          8192
```

Note: The preceding example shows only a portion of the screen. △

Impact Analysis and Reverse Impact Analysis Windows

Impact analysis identifies the tables, columns, jobs, and transformations that are *affected by* a change in a selected table or column. Reverse impact analysis identifies the tables, columns, jobs, and transformations that *contribute to* the content of a selected table or column. For more information about these windows, see Chapter 14, “Using Impact Analysis,” on page 255.

Import and Export Metadata Wizards

SAS Data Integration Studio provides the following wizards for the import and export of SAS metadata (metadata in SAS Open Metadata Architecture format):

Table 2.4 Wizards for Import and Export of SAS Metadata

Wizard	Description
Export Wizard	Exports SAS metadata to a SAS Package (SPK) file.
Import Wizard	Imports SAS metadata that was exported to a SAS Package (SPK) file.
Transformation Importer	Imports one or more generated transformations that were exported in XML format in SAS Data Integration Studio versions earlier than 3.4.
Job Export and Merge Wizard	Imports a job that was previously exported in XML format in SAS Data Integration Studio versions earlier than 3.4.

SAS Data Integration Studio provides the following wizards for the import and export of *other metadata* (metadata that is not in SAS Open Metadata Architecture format):

Table 2.5 Wizards for Import and Export of Other Metadata

Wizard	Description
Metadata Importer	Imports metadata in Common Warehouse Metamodel (CWM) format or in a format that is supported by a SAS Metadata Bridge. You have the option of comparing the imported metadata to existing metadata. You can view any changes in the Differences window and choose which changes to apply.
Metadata Exporter	Exports the default metadata repository that is specified in your metadata profile. You can export metadata in Common Warehouse Metamodel (CWM) format or in a format that is supported by a SAS Metadata Bridge. If you are not exporting in CWM format, then you must license the appropriate bridge.

For more information about metadata import and export, see Chapter 4, “Importing, Exporting, and Copying Metadata,” on page 65.

Source Designer Wizards

Use a source designer to register one or more tables or external files that exist in physical storage. Source designers use a combination of libraries and servers to access the desired tables or external files. Typically, administrators set up the appropriate resources, and SAS Data Integration Studio users simply select the appropriate library, tables, or files in the source designer.

For more information about using source designers, see "Registering Tables with a Source Designer" on page 85. For details about setting up the libraries and servers for source designers, administrators should see the chapters about common data sources in the *SAS Intelligence Platform: Data Administration Guide*.

Target Table Wizard

The Target Table wizard enables you to register a table that does not yet exist in physical storage. For example, suppose that you want to create a SAS Data Integration Studio job that sends its output to a new table. You can use the Target Table wizard to register that table so that you can add the table to the process flow for the job.

As you specify the metadata for a new table, the Target Table wizard enables you to select column metadata from one or more tables that are registered in the current repository. For example, suppose that you want to register a new table, Table C, and that Table C will have some of the same columns as two registered tables, Tables A and B. You can use the Target Table wizard to copy the desired column metadata from Tables A and B into the metadata for Table C. For more information about using the Target Table wizard, see "Registering Tables with the Target Table Wizard" on page 87.

SAS OLAP Cube Wizards

A SAS OLAP cube is a logical set of data that is organized and structured in a hierarchical, multidimensional arrangement. It is a data store that supports online analytical processing (OLAP). When you specify a cube, you specify the dimensions and measures for the cube along with information about how aggregations should be created and stored.

A cube can be quite complex. Accordingly, someone who is familiar with OLAP design and the business goals for a particular cube should design the cube before you create it in SAS Data Integration Studio. For more information about maintaining cubes, see the cube chapter in *SAS Intelligence Platform: Data Administration Guide*. See also the topic "Maintaining Cubes" in SAS Data Integration Studio help.

Here are the main cube wizards in SAS Data Integration Studio.

Table 2.6 Cube Wizards

Wizard	Description
Cube Designer	Adds and maintains a SAS OLAP cube. For more information, see the Help for this wizard.
Export Cube	Exports a SAS OLAP cube in XML format. For more information, see the Help for this wizard.
Import Cube	Imports a SAS OLAP cube that was exported in XML format. For more information, see the Help for these wizards.

Wizard	Description
Advanced Aggregation Tuning	Adds, edits, and deletes the calculated members associated with the cubes that are registered to a current metadata repository. For more information, see the Help for this wizard.
Calculated Members	Adds, edits, and deletes the calculated members associated with the cubes that are registered to a current metadata repository. For more information, see the Help for this wizard.

Data Surveyor Wizards

You can install optional data surveyor wizards that enable you to extract, search, and navigate data from SAP, Siebel, PeopleSoft, Oracle, and other enterprise application vendors. For details about setting up the libraries, servers, and client software for Enterprise Resource Planning (ERP) systems, administrators should see the chapters about common data sources in the *SAS Intelligence Platform: Data Administration Guide*.

Job Windows and Wizards

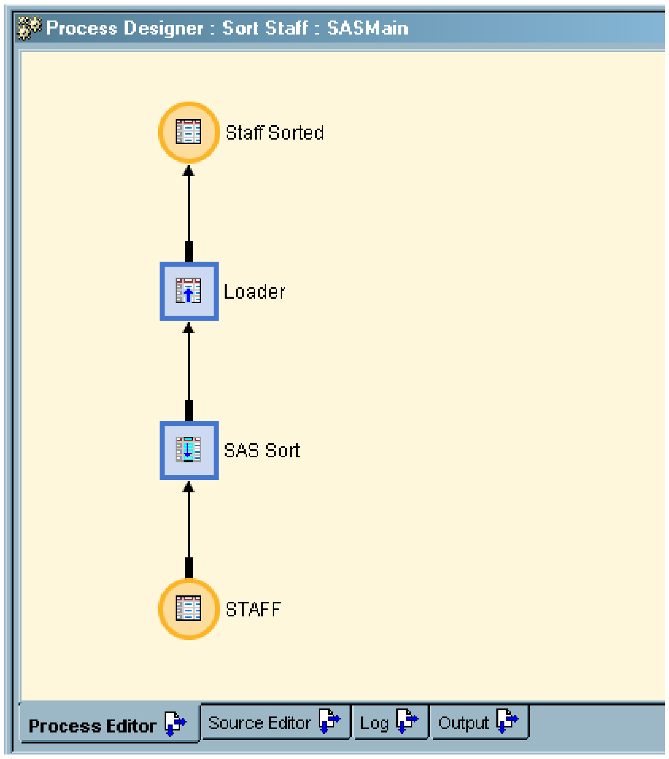
Process Designer Window

Use the Process Designer window to perform these tasks:

- Maintain the process flow diagram for the selected job.
- View or update the metadata for sources, targets, and transformations within the selected job.
- View or update the code that is generated for the entire selected job or for a transformation within that job.
- View a log that indicates whether code was successfully generated for the selected job or for one of its transformations (and was successfully executed, if the code was submitted for execution).
- View any output that the selected job or one of its transformations sends to the SAS output window.

The following display shows a typical view of this window.

Display 2.7 Process Designer Window



In the previous display, the Process Designer window contains the process flow diagram for the Sort Staff job that is described in “Creating a Table That Sorts the Contents of a Source” on page 279. Note that the `Process Editor` tab is shown by default.

The following steps describe one way to open an existing job in the Process Designer window:

1 From the SAS Data Integration Studio desktop, display the Inventory tree.

2 In the Inventory tree, expand the Jobs group.

3 Select the desired job, then select **View ▶ View Job** from the menu bar. The process flow diagram for the job displays in the `Process Editor` tab in the Process Designer window.

If the diagram is too large to view in the `Process Editor` tab, select **View ▶ Overview** from the menu bar. A small image of the complete process flow diagram displays in the Overview window.

To change the size or the orientation of the process flow diagram, select **Process ▶ Zoom** or **Process ▶ Layout** from the menu bar.

The tabs in the Process Designer window are described in the following sections. To display the online Help for each tab, select the tab and press the `F1` key.

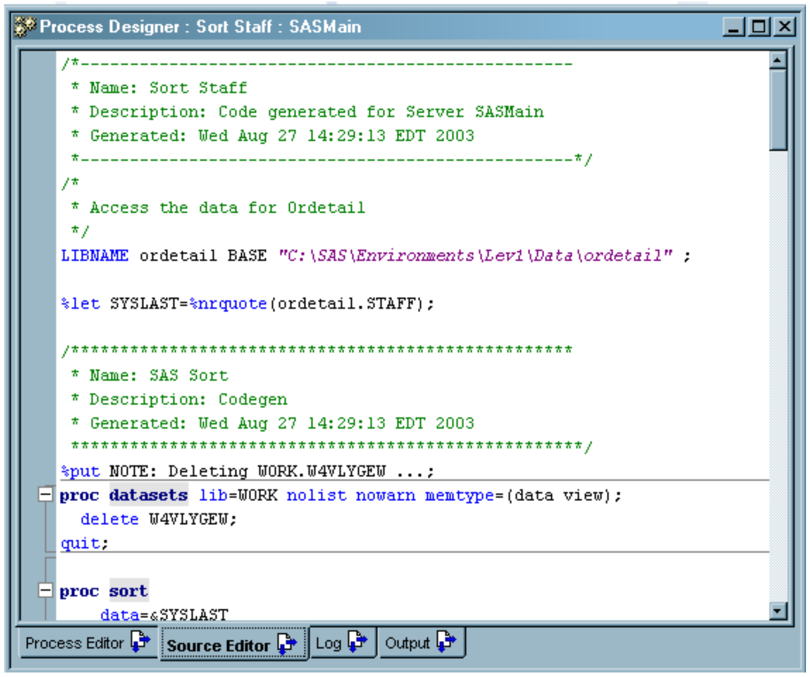
Process Editor Tab

Use the `Process Editor` tab to add and maintain a process flow diagram for the selected job. For an example of how you can use the Process Editor to create a process flow diagram for a job, see “Creating a Process Flow for a Job” on page 144.

Source Editor Tab

Use the **Source Editor** tab to view or modify SAS code for the selected job. For example, if the Sort Staff job is displayed on the **Process Editor** tab, and you select the **Source Editor** tab, code for the entire job is generated and displayed. The following display shows some of the code that would be generated for the Sort Staff job.

Display 2.8 Source Editor Tab



```
/*-------------------------------------------------
 * Name: Sort Staff
 * Description: Code generated for Server SASMain
 * Generated: Wed Aug 27 14:29:13 EDT 2003
 *-------------------------------------------------*/
/*
 * Access the data for Ordetail
 */
LIBNAME ordetail BASE "C:\SAS\Environments\Lev1\Data\ordetail" ;

%let SYSLAST=%nrquote(ordetail.STAFF);

/**************************************************
 * Name: SAS Sort
 * Description: Codegen
 * Generated: Wed Aug 27 14:29:13 EDT 2003
 **************************************************/
%put NOTE: Deleting WORK.W4VLYGEW ...;
proc datasets lib=WORK nolist nowarn memtype=(data view);
  delete W4VLYGEW;
quit;

proc sort
    data=&SYSLAST
```

Log Tab

Use the **Log** tab to view the SAS log that is returned from the code that was submitted for execution. The **Log** tab can help you identify the cause of problems with the selected job or transformation. For information about how you can use the **Log** tab, see “Using SAS Logs” on page 249.

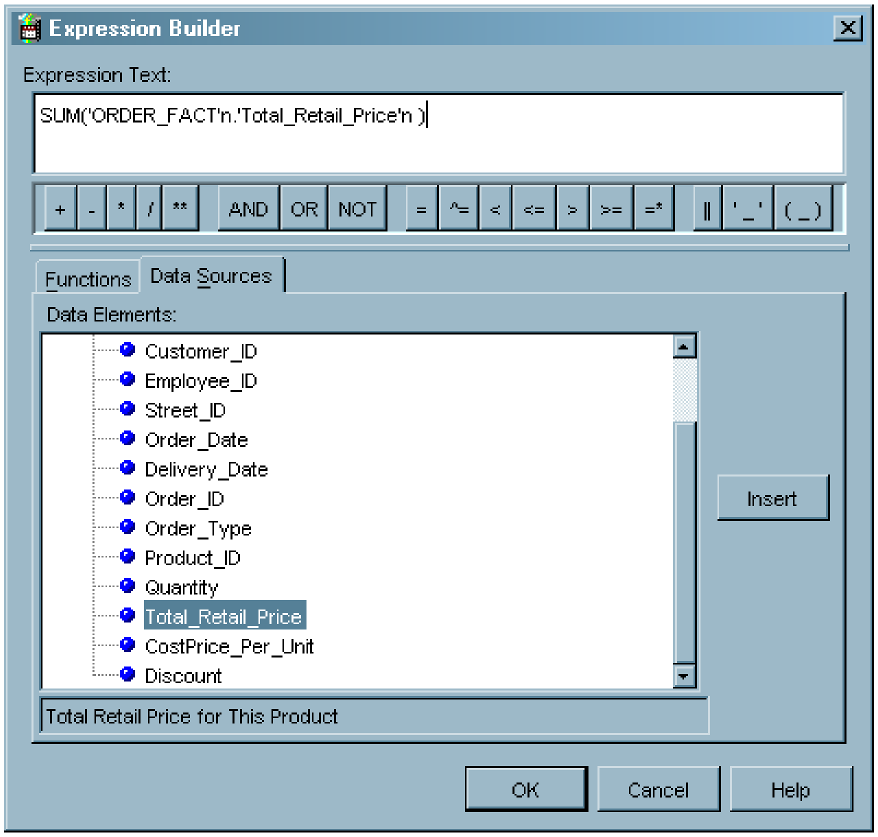
Output Tab

Use the **Output** tab to view any printed output from a SAS program. For example, SAS Data Integration Studio jobs that produce reports can specify that the reports are sent to the **Output** tab. For an example of a job that sends output to the **Output** tab, see “Creating and Using a Generated Transformation” on page 226.

Expression Builder Window

Use the Expression Builder to create SAS expressions that aggregate columns, perform conditional processing, and perform other tasks in a SAS Data Integration Studio job. For example, the following display shows an expression that sums the values in a column named Total_Retail_Price.

Display 2.9 Expression Builder Window with a SUM Expression



```
SUM('ORDER_FACT'n.'Total_Retail_Price'n )
```

Customer_ID
Employee_ID
Street_ID
Order_Date
Delivery_Date
Order_ID
Order_Type
Product_ID
Quantity
Total_Retail_Price
CostPrice_Per_Unit
Discount

Total Retail Price for This Product

The Expression Builder is displayed from tabs in the property windows of many SAS Data Integration Studio transformations. It is used to add or update expressions in SAS, SQL, or MDX. The expression can transform columns, provide conditional processing, calculate new values, and assign new values. The expressions specify the following elements, among others:

- column names
- SAS functions
- constants (fixed values)
- sequences of operands (something to be operated on like a column name or a constant) and operators, which form a set of instructions to produce a value

An expression can be as simple as a constant or a column name, or an expression can contain multiple operations connected by logical operators. For example, an expression to define how the values for the column COMMISSION are calculated can be **`amount * .01`**. An example of conditional processing to subset data can be **`amount > 10000 and region = 'NE'`**. Other examples are an expression to convert a character date into a SAS date or an expression to concatenated columns. For details about SAS expressions, see *SAS Language Reference: Concepts* in SAS OnlineDoc.

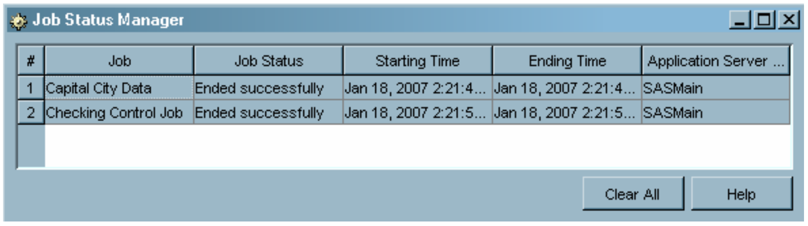
Job Status Manager Window

Use the Job Status Manager window to display the name, status, starting time, ending time, and application server used for all jobs submitted in the current session. Right-click on any row to clear, view, cancel, kill, or resubmit a job.

Note: To debug the SAS code for a job, use the **Log** tab in the Process Designer window. For more information, see “Troubleshooting a Job” on page 155. △

The Job Status Manager window opens in a table that displays all jobs that are submitted in the current session. Each row in the table represents a job in the following example.

Display 2.10 Job Display Manager Window



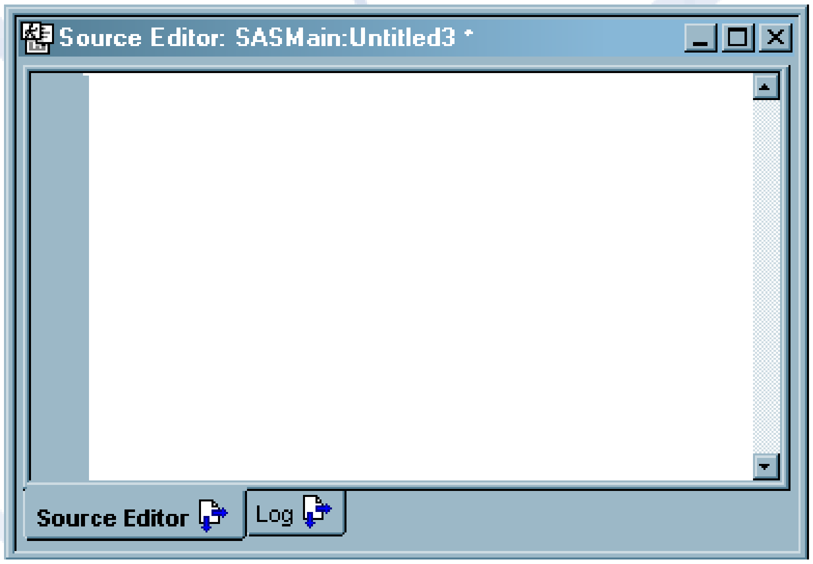
#	Job	Job Status	Starting Time	Ending Time	Application Server ...
1	Capital City Data	Ended successfully	Jan 18, 2007 2:21:4...	Jan 18, 2007 2:21:4...	SASMain
2	Checking Control Job	Ended successfully	Jan 18, 2007 2:21:5...	Jan 18, 2007 2:21:5...	SASMain

For more information, see “Using the Job Status Manager” on page 159.

Source Editor Window

SAS Data Integration Studio also provides a Source Editor window that you can use as a general-purpose SAS code editor. The following display shows an example of this window.

Display 2.11 Source Editor Window



To display the Source Editor window, from the SAS Data Integration Studio desktop, select **Tools** ▶ **Source Editor**.

To submit code from the Source Editor, from the SAS Data Integration Studio desktop, select **Editor** ▶ **Submit**.

To display Help for this window, press the F1 key.

New Job Wizard

Use the New Job wizard to select one or more tables as the targets (outputs) of a job. This wizard can also be used to create an empty job into which you can drag and drop tables and transformation templates. This is the approach that is described in "Creating an Empty Job" on page 143.

Transformation Generator Wizard

One of the easiest ways to customize SAS Data Integration Studio is to create your own generated transformations. Unlike Java-based plug-ins that require software development, generated transformations are created with a wizard.

The Transformation Generator wizard guides you through the steps of specifying SAS code for a generated transformation and saving the transformation to the current metadata repository. After the transformation is saved, it is displayed in the Process Library tree, where it is available for use in any job.

For details about using the Transformation Generator wizard, see "Creating and Using a Generated Transformation" on page 226.

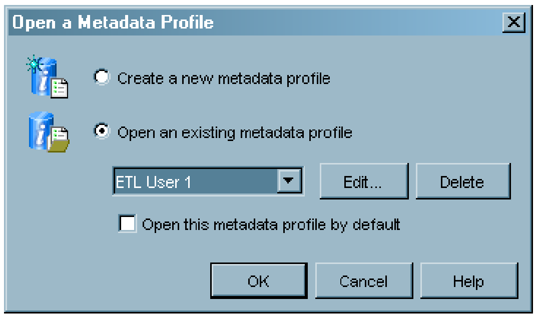
SAS Data Integration Studio Application Windows

Metadata Profile Window

A metadata profile is a client-side definition that specifies the location of a metadata server. The definition includes a host name, a port number, and a list of one or more metadata repositories. In addition, the metadata profile can contain a user's login information and instructions for connecting to the metadata server either automatically or manually.

When you start SAS Data Integration Studio, the Metadata Profile window displays. The following display shows an example of this window.

Display 2.12 Metadata Profile Window


Open a Metadata Profile
Create a new metadata profile
Open an existing metadata profile
ETL User 1
Edit...
Delete
Open this metadata profile by default
OK
Cancel
Help

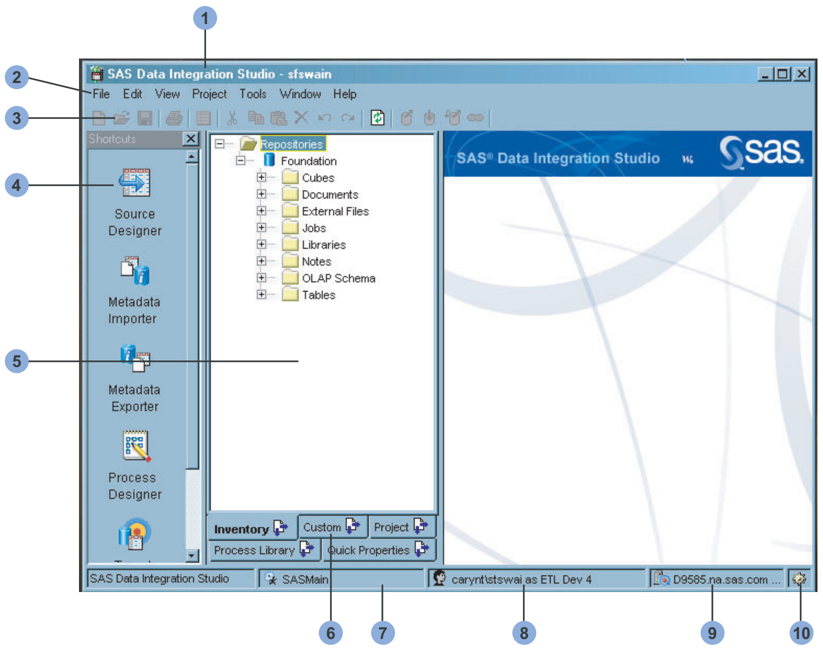
Use the Metadata Profile window to open an existing metadata profile, edit an existing metadata profile, or add a new metadata profile. You must open a metadata profile before you can do any work in SAS Data Integration Studio.

Desktop Window

Overview of the Desktop

After you open a metadata profile, the SAS Data Integration Studio desktop displays.

Display 2.13 SAS Data Integration Studio Desktop



The desktop consists of the following main parts:

1 Metadata profile name
2 Menu bar
3 Toolbar
4 Shortcut bar
5 Tree view
6 Tree tabs
7 Default SAS Application Server
8 User ID and identity
9 Metadata server and port
10 Job status icon

Metadata Profile Name

The title bar in this window shows the name of the metadata profile that is currently in use. For more information about metadata profiles, see "Connecting to a Metadata Server" on page 51.

Menu Bar

Use the menu bar to access the drop-down menus. The list of active options varies according to the current work area and the kind of object that you select. Inactive options are disabled or hidden.

Toolbar

The toolbar contains shortcuts for items on the menu bar. The list of active options varies according to the current work area and the kind of object that you select. Inactive options are disabled or hidden.

Default SAS Application Server

This pane displays the default SAS Application Server, if you have selected one. The message **`No Default Application Server`** indicates that you need to select a default SAS Application server. If you double-click this pane, the Default SAS Application Server window opens. From that window, you can select a default server or test your connection to the default server. Alternatively, you can select the default SAS Application Server from the Options window, as described in “Options Window” on page 34. For more information about the impact of the default SAS Application Server, see “SAS Application Servers” on page 6.

User ID and Identity

This pane displays the domain, login user ID, and metadata identity that are specified in the current metadata profile.

Shortcut Bar

The shortcut bar displays a pane of task icons on the left side of the SAS Data Integration Studio desktop. To display it, select **View ▶ Shortcut Bar** from the menu bar. Each icon displays a commonly used window, wizard, or a selection window for wizards.

Metadata Server and Port

This pane displays the name and port of the SAS Metadata Server that is specified in the current metadata profile.

Job Status Icon

You can right-click this icon to display a list of the last five unique jobs that were submitted in the current SAS Data Integration Studio session and the status for each of them.

Note: The appearance of the icon changes to indicate changes in the status of the jobs that you submitted in the current session. The icon appearance returns to normal when you right-click the icon. △

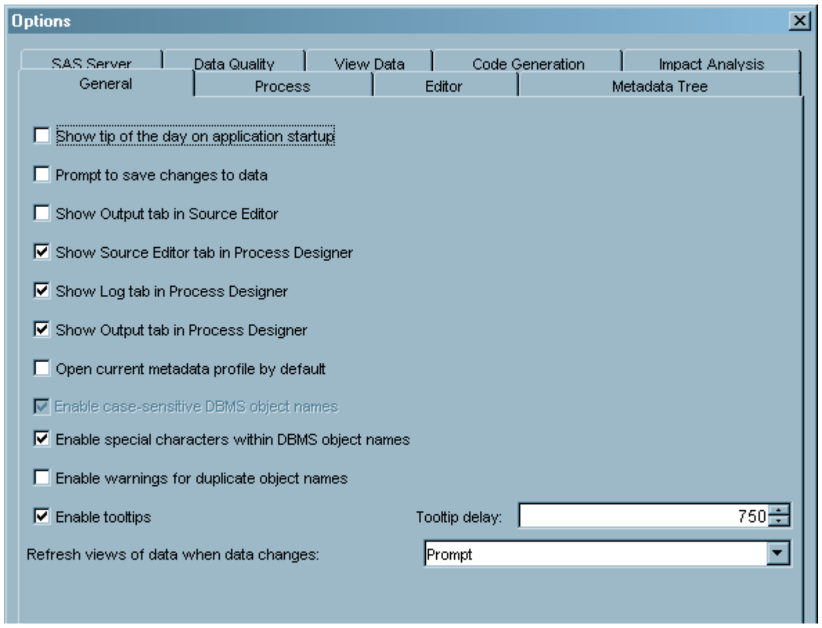
Options Window

Use the Options window to specify global options for SAS Data Integration Studio. The following steps describe one way to display this window:

1. From the SAS Data Integration Studio desktop, select **Tools ▶ Options**.
2. Select the tab that contains the options that you want to view or update.

The following display shows an example of this window.

Display 2.14 Options Window



- [ ] Show tip of the day on application startup
- [ ] Prompt to save changes to data
- [ ] Show Output tab in Source Editor
- [x] Show Source Editor tab in Process Designer
- [x] Show Log tab in Process Designer
- [x] Show Output tab in Process Designer
- [ ] Open current metadata profile by default
- [x] Enable case-sensitive DBMS object names
- [x] Enable special characters within DBMS object names
- [ ] Enable warnings for duplicate object names
- [x] Enable tooltips

Tooltip delay: 750

Refresh views of data when data changes: Prompt

The Options window contains the tabs listed in the following table.

Table 2.7 Option Window Tabs

Tab	**Description**
General	Specifies general user interface options for SAS Data Integration Studio, such as desktop colors or whether to show the **`SAS Log`** tab in the Process Designer.
Process	Specifies options for the **`Process Editor`** tab in the Process Designer. These options control how process flow diagrams display.
Editor	Specifies options for the appearance and behavior of the **`Source Editor`** tab in the Process Designer window.
Metadata Tree	Specifies what types of metadata are displayed in the Metadata tree on the SAS Data Integration Studio desktop.
SAS Server	Specifies the default SAS Application Server for SAS Data Integration Studio.
Data Quality	Specifies options for Data Quality transformation templates that are available in the Process Library tree.
View Data	Specifies options for the View Data window that is available for tables, external files, temporary output tables displayed in process flow diagrams, and selected transformations.

Tab	Description
Impact Analysis	Specifies whether impact analysis extends to dependent metadata repositories.
Code Generation	Specifies options for code generation and parallel processing.

Tree View on the Desktop

Tree View

The tree view is the pane on the left side of the SAS Data Integration Studio desktop that displays the contents of the current metadata repositories. Most of the tabs at the bottom of this pane, such as **`Inventory`** and **`Custom`**, are used to display different hierarchical lists or "trees" of metadata. To display or hide a tree, use the **`View`** option on the menu bar.

Inventory Tree

Use the Inventory tree to select metadata from a default folder for each kind of metadata. For example, you can select table metadata from the folder named **`Tables`**. You can select job metadata from the folder named **`Jobs`**, and so on. The Inventory tree displays the objects in the default metadata repository, as well as the objects from any repositories on which the default repository depends.

The following table describes the main icons for metadata objects in the Inventory tree.

Table 2.8 Main Icons for Metadata Objects in the Inventory Tree

Folder Name	Icon	Description
Repositories	🛢	Icon for a SAS Metadata Repository. A metadata repository is collection of related metadata objects, such as the metadata for a set of tables and columns that are maintained by an application.
Cubes	◇	Metadata for a SAS cube, a logical set of data that is organized and structured in a hierarchical, multidimensional arrangement. See "SAS OLAP Cube Wizards" on page 26.
Deployed Jobs	▦	Metadata for a SAS Data Integration Studio job that has been deployed for scheduling. See "About Job Scheduling" on page 170.
Documents	📄	Metadata for a document. Use the **`Notes`** tab in the properties window for a job, table, column, or another object to associate a document with that object. Unlike notes, which are plain text, a document can contain graphics as well as text.

Folder Name	Icon	Description
External Files		Metadata for an external file. An external file is a file that is created and maintained by a host operating system or by another vendor's software application. A comma-delimited file is one example. See Chapter 6, "Working with External Files," on page 119.
Generated Transforms		Metadata for a transformation that is created with the Transformation Generator wizard. The wizard helps you specify SAS code for the transformation. See "Creating and Using a Generated Transformation" on page 226.
Information Maps		Metadata for an Information Map. Information Maps are created and maintained in SAS Information Map Studio. In SAS Data Integration Studio, you can run impact analysis on Information Maps.
Jobs		Metadata for a SAS Data Integration Studio job. A job is collection of SAS tasks that create output. See "About Jobs" on page 142.
Libraries		Metadata for a library. In SAS software, a library is a collection of one or more files that are recognized by SAS and that are referenced and stored as a unit. See "Libraries" on page 8.
Message Queues		Metadata for a message queue. A message queue is a place where one program can send messages that will be retrieved by another program. See "About Message Queues" on page 361.
Mining Results		Metadata for the output of a Mining Results transformation.
Notes		Metadata for a note. Use the **`Notes`** tab in the properties window for a job, table, column, or another object to associate a note with that object. Unlike documents, which can contain graphics as well as text, notes can contain text only. See "Add and Maintain Notes and Documents for a Column" on page 103.
OLAP Schema		Metadata for an OLAP schema.
Stored Processes		Metadata for a stored process that was generated from a SAS Data Integration Studio job. Enables users to execute jobs from applications such as SAS Enterprise Guide or a Web Service client. See "About SAS Stored Processes" on page 177.
Tables		Metadata for a table. See Chapter 5, "Working with Tables," on page 83.

The following table lists modifier icons that are used in combination with the main icons for folder objects. A modifier icon indicates that an object is in a certain state or has special attributes.

Table 2.9 Modifier Icons for Objects in the Inventory Tree

Folder Name	Icon	Description
Deployed Jobs		A job icon with a clock indicates the job has been deployed for scheduling. See "About Job Scheduling" on page 170.
Jobs		A job icon with a blue triangle indicates that the job has been deployed for scheduling or for execution as a stored process. If the job is deployed for scheduling, then the deployed job appears with a clock icon in the Deployed Jobs folder of the Inventory tree, as shown in the previous table. See "About Job Scheduling" on page 170. If the job is deployed for execution as a stored process, then the deployed job appears with a stored process icon in the Stored Processes folder of the Inventory tree, as shown in the previous table. See "About SAS Stored Processes" on page 177.
various		An icon with an ampersand indicates that some attributes of the object, such as its physical path, are specified as variables rather than literal values. For example, parameterized tables and jobs are often used in iterative jobs. See "Creating a Parameterized Job" on page 335.
Jobs		A job with both an ampersand and a blue triangle indicates that the job specifies its inputs, outputs, or both as parameters. It also indicates that the job has been deployed for scheduling or for execution as a stored process.
various		An icon with a red check mark indicates that this metadata object has been checked out for updating, under change management. Change management enables multiple SAS Data Integration Studio users to work with the same metadata repository at the same time without overwriting each other's changes. See "Working with Change Management" on page 59.

Custom Tree

Use the Custom tree to group related metadata objects together. For example, you can create a folder named Sales to contain the metadata for libraries, tables, and jobs that contain sales information. The Custom tree displays the objects in the default metadata repository, as well as the objects from any repositories on which the default repository depends. The Custom tree has the same folder structure as the tree for the BI Manager plug-in for SAS Management Console. In the Custom tree, the main folder of interest is named **`SAS Data Integration Studio Custom Tree`**. The other folders (**`BIP Tree`**, **`Integration Technologies`**, and **`Samples`**) contain metadata objects from other applications.

Process Library Tree

Overview of the Process Library

The Process Library tree organizes transformations into a set of folders. You can drag a transformation from the Process Library tree and into the Process Editor, where you can populate the drop zones and update the default metadata for the transformation. By updating a transformation with the metadata for actual sources, targets, and transformations, you can quickly create process flow diagrams for common scenarios.

The following sections describe the contents of the Process Library folders.

Access Folder

The following table describes the transformations in the **Access** folder in the Process Library.

Table 2.10 Access Folder Transformations

Name	Description
File Reader	Reads an external file and writes to a target. Added automatically to a process flow when an external file is specified as a source. Executes on the host where the external file resides, as specified in the metadata for the external file.
File Writer	Reads a source and writes to an external file. Added automatically to a process flow when an external file is specified as a target. Executes on the host where the external file resides, as specified in the metadata for the external file.
Library Contents	Creates a control table to use with an iterative job, a job with a control loop in which one or more processes are executed multiple times.
Microsoft Queue Reader	Delivers content from a Microsoft MQ message queue to SAS Data Integration Studio. If the message is being sent into a table, the message queue content is sent to a table or a Data Integration Studio transformation. If the message is being sent to a macro variable or file, then these files or macro variables can be referenced by a later step.
Microsoft Queue Writer	Enables writing files in binary mode, tables, or structured lines of text to the WebSphere MQ messaging system. The queue and queue manager objects necessary to get to the messaging system are defined in SAS Management Console.
SPD Server Table Loader	Reads a source and writes to a SAS SPD Server target. It is automatically added to a process flow when a SAS SPD Server table is specified as a source or as a target. Enables you to specify options that are specific to SAS SPD Server tables.
Table Loader	Reads a source table and writes to a target table. Added automatically to a process flow when a table is specified as a source or a target.
Websphere Queue Reader	Delivers content from a WebSphere MQ message queue to SAS Data Integration Studio. If the message is being sent into a table, the message queue content is sent to a table or a Data Integration Studio transformation. If the message is being sent to a macro variable or file, then these files or macro variables can be referenced by a later step.

Name	Description
Websphere Queue Writer	Enables writing files in binary mode, tables, or structured lines of text to the WebSphere MQ messaging system. The queue and queue manager objects necessary to get to the messaging system are defined in SAS Management Console.
XML Writer	Puts data into an XML table. In a SAS Data Integration Studio job, if you want to put data into an XML table, you must use an XML Writer transformation. You cannot use the Table Loader transformation to load an XML table, for example.

Analysis Folder

The following table describes the transformations in the **`Analysis`** folder in the Process Library.

Table 2.11 Analysis Folder Transformations

Name	Description
Correlations	Creates an output table that contains correlation statistics.
Correlations - Report	Creates an HTML report that contains summary correlation statistics.
Distribution Analysis	Creates an output table that contains a distribution analysis.
Distribution Analysis - Report	Creates an HTML report that contains a distribution analysis.
Forecasting	Generates forecasts for sets of time-series or transactional data.
Frequency	Creates an output table that contains frequency information.
Frequency - Report	Creates an HTML report that contains frequency information.
Summary Statistics	Creates an output table that contains summary statistics.
Summary Statistics - Report	Creates an HTML report that contains summary statistics.
Summary Tables - Report	Creates an HTML report that contains summary tables.

Archived Transforms Folder

The following table describes the deprecated and archived transformations in the **`Archived Transforms`** folder in the Process Library.

Table 2.12 Archived Transforms Folder Transformations

Name	Description
Fact Table Lookup (archived)	Loads source data into a fact table and translates business keys into generated keys.
SQL Join (archived)	Selects multiple sets of rows from one or more sources and writes each set of rows to a single target. Typically used to merge two or more sources into one target. Can also be used to merge two or more copies of a single source.
Table Loader (archived)	Reads a source table and writes to a target table. Added automatically to a process flow when a table is specified as a source or a target.

Control Folder

The following table describes the transformations in the **`Control`** folder in the Process Library.

Table 2.13 Control Folder Transformations

Name	Description
Loop	Marks the beginning of the iterative processing sequence in an iterative job.
Loop End	Marks the end of the iterative processing sequence in an iterative job.
Return Code Check	Provides status-handling logic at a desired point in the process flow diagram for a job. Can be inserted between existing transformations and removed later without affecting the mappings in the original process flow.

Data Transforms Folder

The following table describes the transformations in the **`Data Transforms`** folder in the Process Library.

Table 2.14 Data Transforms Folder Transformations

Name	Description
Append	Creates a single target table by combining data from several source tables.
Apply Lookup Standardization	Applies scheme data sets to transform source columns.
Create Match Code	Establishes relationships between source rows using match code analysis or cluster analysis.
Data Transfer	Moves data directly from one machine to another. Direct data transfer is more efficient than the default transfer mechanism.
Data Validation	Cleanses data before it is added to a data warehouse or data mart.
Extract	Selects multiple sets of rows from a source and writes those rows to a target. Typically used to create one subset from a source. Can also be used to create columns in a target that are derived from columns in a source.

Name	Description
Fact Table Lookup	Loads source data into a fact table and translates business keys into generated keys.
Key Effective Date	Enables change tracking in intersection tables.
Lookup	Loads a target with columns taken from a source and from several lookup tables.
Mining Results	Integrates a SAS Enterprise Miner model into a SAS Data Integration Studio data warehouse. Typically used to create target tables from a SAS Enterprise Miner model.
SAS Rank	Ranks one or more numeric column variables in the source and stores the ranks in the target.
SAS Sort	Reads data from a source, sorts it, and writes the sorted data to a target.
SAS Splitter	Selects multiple sets of rows from one source and writes each set of rows to a different target. Typically used to create two or more subsets of a source. Can also be used to create two or more copies of a source.
SCD Type 2 Loader	Loads source data into a dimension table, detects changes between source and target rows, updates change tracking columns, and applies generated key values. This transformation implements slowly changing dimensions.
SQL Join	Selects multiple sets of rows from one or more sources and writes each set of rows to a single target. Typically used to merge two or more sources into one target. Can also be used to merge two or more copies of a single source.
Standardize	Creates an output table that contains data standardized to a particular number.
Surrogate Key Generator	Loads a target, adds generated whole number values to a surrogate key column and sorts and saves the source based on the values in the business key column(s).
Transpose	Creates an output table that contains transposed data.
User Written Code	Retrieves a user-written transformation. Can be inserted between existing transformations and removed later without affecting the mappings in the original process flow. Can also be used to document the process flow for the transformation so that you can view and analyze the metadata for a user-written transformation, similarly to how you can for other transformations.

Output Folder

The following table describes the transformations in the **`Output`** folder in the Process Library.

Table 2.15 Output Folder Transformations

Name	Description
List Data	Creates an HTML report that contains selected columns from a source table. See also "Add the List Data Transformation to a Process Flow" on page 253.

Publish Folder

The following table describes the transformations in the **`Publish`** folder in the Process Library.

Table 2.16 Publish Folder Transformations

Name	Description
Publish to Archive	Creates an HTML report and an archive of the report.
Publish to Email	Creates an HTML report and e-mails it to a designated address.
Publish to Queue	Creates an HTML report and publishes it to a queue using MQSeries.

SPD Server Dynamic Cluster Folder

The following table describes the transformations in the **`SPD Server Dynamic Cluster`** folder in the Process Library.

Table 2.17 SPD Server Dynamic Cluster Folder Transformations

Name	Description
Create or Add to a Cluster	Creates or updates an SPD Server cluster table.
List Cluster Contents	Lists the contents of an SPD Server cluster table.
Remove Cluster Definition	Deletes an SPD Server cluster table.

Additional Information about the Process Library Transformations

Perform the following steps to display examples that illustrate how the Process Library templates can be used in a SAS Data Integration Studio job:

1 From the SAS Data Integration Studio menu bar, select **Help ▶ Contents**. The online Help window displays.

2 In the left pane of the Help window, select **Examples ▶ Process Library Examples**.

Project Tree

Use the Project tree to work with metadata in your Project repository. A Project repository is a temporary work area where metadata can be added or updated before it is checked in to a change-managed repository. For more information, see “Working with Change Management” on page 59.

Quick Properties Pane

On the desktop, use the Quick Properties pane to display the main attributes of an object that is selected in a tree view.

On the **`Designer`** tab of the properties window for the SQL Join transformation, use the Quick Properties pane to display the main attributes of an object that is selected in the `Create/Subquery` tab. To display or hide this pane, select or deselect **View ▶ SQL Designer Properties** from the menu bar. On this `Designer` tab, you can also use the Quick Properties pane to update the properties for some objects.

Comparison Results Tree

The Comparison Results tree displays the metadata objects that result from change analysis operations. You can select a comparison result object and view the comparison in the Differences window, recompare the specified metadata, and perform other tasks. For more information, see "Working with Other Metadata" on page 75.

Metadata Tree

Use the Metadata tree to identify the metadata type associated with a particular object, as defined in the SAS Open Metadata Architecture. This can be useful in some situations. For example, after identifying the metadata type associated with an object, a developer can write a program that uses the SAS Open Metadata Interface to read and write the metadata for that kind of object. The Metadata tree displays the objects in the default metadata repository, as well as the objects from any repositories on which the default repository depends.

P A R T 2

General User Tasks

CHAPTER
3

Getting Started

Required Components for SAS Data Integration Studio

Certain servers, metadata repositories and other components must be put in place in order for SAS Data Integration Studio to operate properly. The most important components are listed in the following table.

Table 3.1 Required Components

Component	**Description**
Data Integration Environment	In most cases, administrators install and configure servers, metadata repositories, and other elements of a SAS data integration environment, and SAS Data Integration Studio users are told which of these resources to use. These elements are listed in the following table. For more information, see “A Basic Data Integration Environment” on page 4.
Hot Fixes	At least one hot fix must be applied to the metadata servers in order to fully support SAS Data Integration Studio 3.4. Other hot fixes might be needed to support certain features. For more information, administrators should see the installation instructions for SAS Data Integration Studio 3.4.
SAS Metadata Server Version and Hot Fixes	The improved import/export and copy/paste features in SAS Data Integration Studio 3.4 require a SAS 9.1.3 Metadata Server with Service Pack 4, as well as a hot fix. The hot fix is applied once per metadata server.

Component	Description
SAS Workspace Server Version and Hot Fixes	SAS Workspace Servers execute SAS Data Integration Studio jobs and are also used to access data. A SAS 9.1.3 Workspace Server with Service Pack 3 will support all features in SAS Data Integration Studio 3.4 except for Websphere message queues. If you will be using Websphere message queues, administrators should apply the appropriate hotfix to the SAS Workspace Server that will execute jobs that read output from, or send output to, Websphere message queues.
Metadata Updates for SAS Data Integration Studio	SAS Data Integration Studio 3.4 requires metadata that must be applied by SAS Management Console 9.1 with hot fixes for SAS Data Integration Studio 3.4. Administrators should perform the following steps once per metadata server. Repeat for all metadata servers that SAS Data Integration Studio users will access. 1 Open SAS Management Console as an unrestricted user (such as sasadm). 2 From the desktop, in the Repository selection panel, select the relevant Foundation repository. 3 Select **Tools ▶ Update Metadata for SAS Data Integration Studio** from the menu bar. The **`Update Metadata for SAS Data Integration Studio`** dialog box displays. 4 Click **`OK`** to update the metadata for the current metadata server. *Note:* It is no longer necessary to restart the metadata server so that the updates can take effect. △

Main Tasks for Creating Process Flows

When you create process flows in SAS Data Integration Studio, perform the following main tasks:

1. Start SAS Data Integration Studio. See “Starting SAS Data Integration Studio” on page 50.
2. Create and open a metadata profile. See “Connecting to a Metadata Server” on page 51.
3. Select a default SAS Application Server. See “Selecting a Default SAS Application Server” on page 55.
4. Add metadata for the job’s inputs (data sources). See “Registering Tables with a Source Designer” on page 85.
5. Add metadata for the job’s outputs (data targets). See “Registering Tables with the Target Table Wizard” on page 87.
6. Create a new job and a process flow that will read the appropriate sources, perform the required transformations, and load the target data store with the desired information. See Chapter 7, “Creating, Executing, and Updating Jobs,” on page 141.
7. Run the job. See “Submitting a Job for Immediate Execution” on page 147.

The change management feature adds a few steps to some of the previous tasks. For more information, see “Working with Change Management” on page 59.

Starting SAS Data Integration Studio

Problem

You want to start SAS Data Integration Studio and perhaps specify one or more start options.

Solution

Start SAS Data Integration Studio as you would any other SAS application on a given platform. You can specify one or more options in the start command or in the **`etlstudio.ini`** file.

Tasks

Start Methods

Under Microsoft Windows, you can select **Start ▶ Programs ▶ SAS ▶ SAS Data Integration Studio**.

You can also start the application from a command line. Navigate to the SAS Data Integration Studio installation directory and issue the **`etlstudio.exe`** command.

If you do not specify any options, SAS Data Integration Studio uses the parameters specified in the **`etlstudio.ini`** file. The following sections contain information about options you can specify on the command line or add to the **`etlstudio.ini`** file.

Specify Java Options

To specify Java options when you start SAS Data Integration Studio, use the **`-javaopts`** option and enclose the Java options in single quotation marks. For example, the following command starts SAS Data Integration Studio on Windows and contains Java options that specify the locale as Japanese:

```
etlstudio -javaopts '-Duser.language=ja -Duser.country=JP'
```

Specify the Plug-in Location

By default, SAS Data Integration Studio looks for plug-ins in a **`plugins`** directory under the directory in which the application was installed. If you are starting SAS Data Integration Studio from another location, you must specify the location of the plug-in directory by using the **`--pluginsDir`** option. The syntax of the option is

```
etlstudio --pluginsdir <plugin path>
```

Specify the Error Log Location

SAS Data Integration Studio writes error information to a file named **`errorlog.txt`** in the working directory. Because each SAS Data Integration Studio session overwrites this log, you might want to specify a different name or location for the log file. Use the following option to change the error logging location:

```
etlstudio --logfile '<filepath/filename>'
```

Specify Message Logging

You can specify the server status messages that are encountered in a SAS Data Integration Studio session by using the **`--MessageLevel`** ***`level_value`*** option. Valid values for *level_value* are listed in the following table.

Table 3.2 Values for level_value

Value	Description
ALL	All messages are logged.
CONFIG	Static configuration messages are logged.
FINE	Basic tracing information is logged.
FINER	More detailed tracing information is logged.
FINEST	Highly detailed tracing information is logged. Specify this option to debug problems with SAS server connections.
INFO	Informational messages are logged.
OFF	No messages are logged.
SEVERE	Messages indicating a severe failure are logged.
WARNING	Messages indicating a potential problem are logged.

Allocate More Memory to SAS Data Integration Studio

There might be a number of reasons to increase the amount of memory for SAS Data Integration Studio. For example, after running a job, if you click the **`Log`** tab or the **`Output`** tab, and SAS Data Integration Studio does not respond, you might need to increase the amount of memory allocated to the application.

Locate the subdirectory where SAS Data Integration Studio's executable (**`etlstudio.exe`**) is found. There will be a .ini file with the same name as the executable (**`etlstudio.ini`**). Edit the .ini file and increase the memory values on the Java invocation. If that does not help, the problem might be server memory or another issue.

Connecting to a Metadata Server

Problem

You want to connect to the metadata server where the metadata for the required servers, data libraries, and tables is stored.

Solution

Create and open a metadata profile that connects to the appropriate metadata server and repository.

A metadata profile is a client-side definition of where the metadata server is located. The definition includes a machine name, a port number, and one or more metadata repositories. In addition, the metadata profile can contain a user's logon information and instructions for connecting to the metadata server automatically. A metadata profile is stored in a file that is local to the SAS Data Integration Studio client. It is not stored in a metadata repository.

Tasks

Create a Profile for a User, without Change Management

Perform the following steps to create a metadata profile that will enable you to work with a metadata repository that is not under change-management control. Assume that you are not an administrator.

1 Obtain the following information from an administrator:
   - the network name of the metadata server
   - the port number used by the metadata server
   - a logon ID and password for that server
   - the name of the repository that should be selected as the default metadata repository for this profile.

2 Start SAS Data Integration Studio. The Open a Metadata Profile window displays.

3 Select `Create a new metadata profile`. The Metadata Profile wizard displays.

4 Click `Next`. In the general information window, enter a name for the profile.

5 Click `Next`. In the Connection Information window, enter a machine address, port, user name, and password that will enable you to connect to the appropriate SAS Metadata Server.

6 Click `Next`. The wizard attempts to connect to the metadata server. If the connection is successful, the Select Repositories window displays.

7 In the Select Repositories window, select the appropriate repository as the default metadata repository for this profile.

8 Click `Finish` to exit the metadata profile wizard. You are returned to the Open a Metadata Profile window.

Create a Profile for a User, with Change Management

The change management feature in SAS Data Integration Studio enables multiple users to work simultaneously with a shared metadata repository. For more information about this feature, see “Working with Change Management” on page 59.

Perform the following steps to create a metadata profile that will enable you to work with a metadata repository that is under change-management control. Assume that you are not an administrator and that administrators have set up change-managed repositories.

1 Obtain the following information from an administrator:
   - the network name of the metadata server
   - the port number used by the metadata server
   - a logon ID and password for that server
   - the name of the Project repository that should be selected as the default metadata repository for this profile. The administrator who created the repository must have specified you as the owner of the repository.

2 Start SAS Data Integration Studio. The Open a Metadata Profile window displays.
3 Select **`Create a new metadata profile`**. The Metadata Profile wizard displays.
4 Click **`Next`**. In the general information window, enter a name for the profile.
5 Click **`Next`**. In the Connection Information window, enter a machine address, port, user name, and password that will enable you to connect to the appropriate SAS Metadata Server.
6 Click **`Next`**. The wizard attempts to connect to the metadata server. If the connection is successful, the Select Repositories window displays.
7 In the Select Repositories window, select the appropriate Project repository as the default metadata repository for this profile.
8 Click **`Finish`** to exit the metadata profile wizard. You are returned to the Open a Metadata Profile window.

Create a Profile for an Administrator, with Change Management

Some tasks, such as deploying a job for scheduling or customizing status handling, cannot be done under change management control. Perform the following steps to create a metadata profile that will grant an administrator special privileges in a change-managed repository:

1 Obtain the following information:
   - the network name of the metadata server
   - the port number used by the metadata server
   - an administrative logon ID and password for that server
   - the name of the change-managed repository that should be selected as the default metadata repository for this profile.

2 Start SAS Data Integration Studio. The Open a Metadata Profile window displays.
3 Select **`Create a new metadata profile`**. The Metadata Profile wizard displays.
4 Click **`Next`**. In the general information window, enter a name for the profile.
5 Click **`Next`**. In the Connection Information window, enter a machine address, port, user name, and password that will enable you to connect to the appropriate SAS Metadata Server.
6 Click **`Next`**. The wizard attempts to connect to the metadata server. If the connection is successful, the Select Repositories window displays.
7 In the Select Repositories window, select the change-managed repository as the default metadata repository for this profile.
8 Click **`Finish`** to exit the metadata profile wizard. You are returned to the Open a Metadata Profile window.

Open a Metadata Profile

Perform the following steps to open a metadata profile:

1 Start SAS Data Integration Studio. The Open a Metadata Profile window displays.
2 Select **`Open an existing metadata profile`**. The selected profile is opened in SAS Data Integration Studio.

Another way to open a metadata profile is to start SAS Data Integration Studio, then select **File ► Open a Metadata Profile** from the menu bar.

After you open a metadata profile, the tree views on the desktop will be populated with metadata that is associated with the default repository that is specified in the profile. If you are working under change management control, you will see a **`Project`** tab in the tree view. If you are not working under change management, you will not see a **`Project`** tab. For more information about change management, see “Working with Change Management” on page 59.

Additional Information

Impact of the Default Repository

When you create a metadata profile, you specify a default metadata repository for that profile. The administrator who creates metadata repositories typically tells SAS Data Integration Studio users what repository to select as the default.

The default metadata repository has two main impacts:

- It determines which metadata objects are available in a SAS Data Integration Studio session. Wizards and the tree views display metadata objects from the default metadata repository and from any repository on which the default repository "depends" (inherits metadata from).
- New metadata objects are always saved to the default metadata repository that is specified in the current profile.

Reconnecting to a Metadata Server

Problem

You are working in SAS Data Integration Studio, and the connection to the metadata server is broken. Any metadata that you have entered but not saved could be lost unless connection to the metadata server is restored.

Solution

If the connection to the metadata server is broken, a dialog box displays and asks if you want to attempt reconnection. Click **`Try Now`**, and SAS Data Integration Studio will attempt to reconnect to the metadata server.

If the reconnection is successful, you can continue your work. The user credentials from the previous session will be used. If the tree views are not populated with the appropriate metadata, select **View ▶ Refresh**. If the reconnection is not successful, contact your server administrator.

Selecting a Default SAS Application Server

Problem

You want to work with SAS Data Integration Studio without having to select a server each time that you access data, execute SAS code, or perform other tasks that require a SAS server.

Solution

Use the **Tools ▶ Options** window to select a default SAS Application Server.

When you select a default SAS Application Server, you are actually selecting a metadata object that can provide access to a number of servers, libraries, schemas, directories, and other resources. An administrator typically creates this object. The administrator then tells the SAS Data Integration Studio user which object to select as the default server.

Tasks

Select a SAS Application Server

Perform the following steps to select a default SAS Application Server:

1 From the SAS Data Integration Studio menu bar, select **Tools ▶ Options** to display the Options window.
2 Select the **`SAS Server`** tab.
3 On the **`SAS Server`** tab, select the desired server from the Server drop-down list. The name of the selected server appears in the **`Server`** field.
4 Click **`Test Connection`** to test the connection to the SAS Workspace Server(s) that are specified in the metadata for the server. If the connection is successful, go to the next step. If the connection is not successful, contact the administrator who defined the server metadata for additional help.
5 After you have verified the connection to the default SAS Application Server, click **`OK`** to save any changes. The server that is specified in the **`Server`** field is now the default SAS Application Server.

Note: For more information about the impact of the default SAS Application Server, see "Accessing Local and Remote Data" on page 148. △

Registering Any Libraries That You Need

Problem

You want to register a library before you can access some tables in that library.

Solution

Use the New Library wizard to register the library.

At some sites, an administrator adds and maintains most of the libraries that are needed, and the administrator tells users which libraries to use. It is possible, however, that SAS Data Integration Studio users will need to register additional libraries.

Tasks

Register a Library

Perform the following steps to register a library:

1 Start SAS Data Integration Studio and open an appropriate metadata profile. The default metadata repository for this profile will store metadata about the library that you will register.

2 From the SAS Data Integration Studio desktop, in the Inventory tree, select the `Libraries` folder, then select **File ▶ New** from the menu bar. The New Library Wizard displays. The first page of the wizard enables you to select the kind of library that you want to create.

3 After you have selected the library type, click **OK**.

4 Enter the rest of the library metadata as prompted by the wizard.

Note: Registering a library does not, in itself, provide access to tables in the library. △

For more information about libraries, see the chapters about common data sources in the *SAS Intelligence Platform: Data Administration Guide*.

You must also specify metadata for all tables that you want to access in the library. For more information, see "Registering Sources and Targets" on page 56.

Registering Sources and Targets

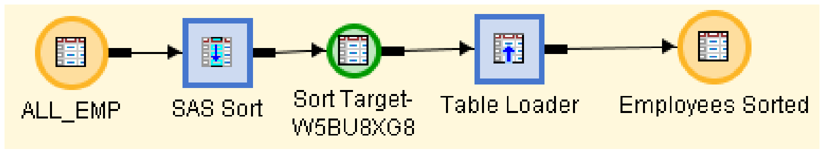
Problem

You want to create a process flow in SAS Data Integration Studio, but the sources and targets are not visible in the tree view.

Solution

Use wizards in SAS Data Integration Studio to register the sources and targets in the desired process flow. For example, suppose that you wanted to create the process flow that is shown in the following display.

Display 3.1 Sample Job with a Source and a Target



Given the direction of the arrows in the process flow:

- □ ALL_EMP specifies metadata for the source table.
- □ SAS Sort specifies metadata for the sort process, which writes its output to a temporary output table, Sort Target-W5BU8XGB.
- □ Table Loader specifies metadata for a process that reads the output from the previous step and loads this data into a target table.
- □ Employees Sorted specifies metadata for the target table.

Before you could create the previous process flow, you must register the source table (ALL_EMP) and the target table (Employees Sorted). The tables will then be visible in the tree views, and you can drag and drop them into the process flow.

Tasks

Use Wizards to Register Sources and Targets

Use the methods in the following table to enter metadata for both sources and targets in SAS Data Integration Studio jobs.

Table 3.3 Methods for Specifying Metadata for Data Stores

Data Store	Method for Specifying Metadata
A set of tables that are specified in a data model.	Import the data model in CWM format or in a format for which you have the appropriate Meta Integration Model Bridge. See "Working with Other Metadata" on page 75.
One or more SAS tables or DBMS tables that exist in physical storage.	Use a source designer to register the tables. See "Registering Tables with a Source Designer" on page 85. See also the chapters about common data sources in the *SAS Intelligence Platform: Data Administration Guide*.
A table that is specified in a comma-delimited file or another external file.	Use an external file source designer to register the table. See Chapter 6, "Working with External Files," on page 119. See also the external file source designer examples in the Help for SAS Data Integration Studio.
A new table that will be created when a SAS Data Integration Studio job is executed. Or, a new table that reuses column metadata from one or more registered tables.	Use the Target Table designer to register the table. See "Registering Tables with the Target Table Wizard" on page 87.

Data Store	Method for Specifying Metadata
One or more tables that are specified in an XML file.	Various methods can be used. One way is to use the XML source designer. See the chapters about common data sources in the *SAS Intelligence Platform: Data Administration Guide*.
A table that can be accessed with an Open Database Connectivity (ODBC) driver, such as a Microsoft Access table.	Use an ODBC source designer to register the table. See the chapters about common data sources in the *SAS Intelligence Platform: Data Administration Guide*.
A Microsoft Excel spreadsheet.	Various methods can be used. One way is to use the Excel source designer. Another way is to use the ODBC source designer. See the chapters about common data sources in the *SAS Intelligence Platform: Data Administration Guide*.
One or more tables that exist in physical storage and that can be accessed with an Open Database Connectivity (ODBC) driver.	Use an ODBC source designer to register the tables. See the chapters about common data sources in the *SAS Intelligence Platform: Data Administration Guide*.
A table in a format for which you do not have a source designer.	Use the Generic source designer to register the table. See the Generic source designer example in the Help for SAS Data Integration Studio.
Add and maintain a SAS cube.	See the cube topics in the Help for SAS Data Integration Studio.

Register DBMS Tables with Keys

Tables in a database management system often have primary keys, unique keys, and foreign keys.

A *primary key* is one or more columns that are used to uniquely identify a row in a table. A table can have only one primary key. The column(s) in a primary key cannot contain null values.

A *unique key* is also one or more columns that can be used to uniquely identify a row in a table. A table can have one or more unique keys. Unlike a primary key, a unique key can contain null values.

A *foreign key* is one or more columns that are associated with a primary key or unique key in another table. A table might have one or more foreign keys. A foreign key is dependent upon its associated primary or unique key. In other words, a foreign key cannot exist without a primary or unique key.

Note: When registering a DBMS table with foreign keys, if you want to preserve the foreign keys, you register all of the tables that are referenced by the foreign keys. △

For example, suppose that Table 1 had foreign keys that referenced primary keys in Table 2 and Table 3. To preserve the foreign keys in Table 1, you could use the Metadata Importer wizard or a source designer wizard to import metadata for Tables 1, 2, and 3.

Working with Change Management

Problem

You want to manage the needs of the users that have access to your metadata. Specifically, you must ensure that users can access the metadata that they need without compromising the data that other users need. This means that you must have a reliable mechanism for adding and deleting metadata. You also must be able to check metadata in and out as needed.

Solution

You can use the change management feature. The change management feature enables multiple users to work simultaneously with a shared metadata repository. It enables teams of developers to build and maintain process flows without overwriting each other's changes.

To implement change management, an administrator creates a metadata repository and puts it under change-management control. Each SAS Data Integration Studio user then creates and opens a metadata profile which enables him or her to work with the change-managed repository.

To add new metadata under change management, you will add the metadata as usual. When you are finished working with the new metadata, you will check it in to the change-managed repository.

To update existing metadata under change management, you will check out the metadata from the change-managed repository. The metadata is locked so that no one else can update it as long as you have it checked out. When you are finished updating the metadata, you check it in to the change-managed repository. Change management enables you to perform the following tasks:

- set prerequisites for change management
- add new metadata
- check existing metadata out for updates
- check metadata in
- delete metadata
- remove metadata permanently
- clear all metadata from the Project tree
- clear project repositories

Tasks

Review the Prerequisites for Change Management

To implement change management, administrators create one or more change-managed repositories. Then they set up user profiles for all SAS Data Integration Studio users who will be working under change management. For more information, administrators should see the SAS Data Integration Studio chapter in *SAS Intelligence Platform: Desktop Application Administration Guide* . Each user must create and open a metadata profile which enables him or her to work with a change-managed repository. For more information, users should see “Create a Profile for a User, with Change Management” on page 52.

Add New Metadata

You can add metadata about a new library, table, or another resource to a change-managed repository. Perform the following steps:

1 If you have not done so already, open a metadata profile that enables you to work under change management. For more information, see “Create a Profile for a User, with Change Management” on page 52.
2 Add the metadata as usual. For example, to register a table, you can use the methods described in “Registering Sources and Targets” on page 56. New metadata objects will appear in your Project tree on the desktop.

Check Out Existing Metadata for Updates

You can update existing metadata for a library, table, or another resource in a change-managed repository. Perform the following steps:

1 If you have not done so already, open a metadata profile that enables you to work under change management. For more information, see “Create a Profile for a User, with Change Management” on page 52.
2 On the SAS Data Integration Studio desktop, click the **Inventory** tab or the **Custom** tab. The appropriate tree displays.
3 Open the folder for the kind of metadata that you want to check out, such as the **Tables** folder for tables in the Inventory tree.
4 Right-click the metadata that you want to check out and select **Change-Management ▶ Check Out**. You can also left-click the metadata that you want to check out, then go the drop-down menu and select **Project ▶ Check Out**. The metadata is checked out and displays in your Project tree. The Project tree displays the contents of your Project repository.

After you are finished working with the metadata, you can check it in as described in “Check Metadata In” on page 60.

Note: You must check out the metadata objects to be changed before you can update the metadata. As long as you have the metadata checked out, it is locked so that it cannot be updated by another person . If two or more tables share a common object, such as a primary key, a note, or a document, and you check out one of these tables, only you can check out the other tables that share that common object. (Other users cannot access the common object that you have checked out, and the shared object is required to check out a table that uses that object.) △

Check Metadata In

When you have finished adding or updating metadata, you can check it in to your change-managed repository. Perform the following steps:

1 In the Project tree, right-click the Project repository icon and select **Check In Repository**. You can also left-click the Project repository icon, open the drop-down menu, and select **Project ▶ Check In Repository**. The Check In window displays.
2 Enter meaningful comments in the **Name** field (and perhaps in the **Description** field) about the changes that were made to all of the objects that you are about to

check in. The text entered here becomes part of the check in/check out history for all objects that you are checking in. If you do not enter meaningful comments, the check in/check out history is less useful.

3 When you are finished entering comments in the Check In window, click `OK`. All metadata objects that are in the Project repository are checked in.

After check in, any new or updated metadata that was in your Project tree will now be visible to others who are connected to the same metadata repository. You might find it convenient to work with small sets of related objects in the Project tree. That way, it is easier to track the metadata changes you are making.

Note: A checkin operation checks in all metadata objects that are in the Project tree. You cannot check in selected objects and leave other objects in the Project tree. △

Delete Metadata

You can delete selected metadata objects from your Project tree. This would be helpful if you added or updated objects in your temporary work area that you do not want to check in to the change-managed repository.

You can use the `Delete` option to remove metadata objects that have been checked out or remove new metadata objects that have never been checked in. New metadata objects are simply deleted. Metadata objects that are checked out are deleted, and the corresponding objects in the change-managed repository are restored to the state they were in before check out. Alternatively, you can use the `Undo Checkout` option to remove metadata objects that have been checked out. It has the same effect as the `Delete` option.

Perform the following steps to delete selected metadata from your Project tree.

1 In the Project tree, select one or more objects that you want to remove.
2 Right-click the object or objects and select `Delete`. Alternatively, for checked-out objects only, you can right-click the objects and select **Change Management ▶ Undo Checkout**. The objects are removed from the Project tree.

Remove Metadata Permanently

You can permanently remove selected metadata objects from the change-managed repository. This ability would be helpful if you added or updated some metadata objects, checked them in, and then discovered that you do not need them. Perform the following steps:

1 Check out the objects to be removed, as described in “Check Out Existing Metadata for Updates” on page 60.
2 Select the objects to be removed from the Project tree.
3 Right-click the objects and select **Change Management ▶ Destroy**.
4 Click `Yes` in response to the dialog box. The objects are removed from the Project tree.
5 Check in the Project repository, as described in “Check Metadata In” on page 60. The objects are permanently removed from the change-managed repository

Note: Metadata objects that are destroyed cannot be recovered except by restoring the change-managed repository from backup. △

Clear All Metadata From the Project Tree

You can clear all metadata from your Project tree. You could do this to remove metadata objects that fail to check in due to technical problems. You perform this task with the `Clear Project Repository` option, which deletes all new objects and unlocks all checked-out objects in the Project tree. When you clear a Project repository, all changes that have not been checked in are lost. Perform the following steps:

1 If you have not done so already, click the `Project` tab. The Project tree displays.
2 Select **Project ▶ Clear Project Repository** from the menu bar. A warning displays that any changes to the current objects will be lost if you click `OK`.
3 Click `OK`. All new objects are deleted. All checked-out objects are unlocked. All changes that have not been checked in are lost.

Clear Project Repositories That You Do Not Own

An administrator might need to clear all metadata from one or more Project repositories that he or she does not own. This would be helpful in the following situations:

- A user checked out metadata objects but forgot to check them back in before going on a long vacation. In the meantime, other users need to update the checked-out metadata.
- An administrator accidentally deletes a user's Project repository that contains checked-out objects.

You can use the `Clear Project Repository` option to delete all new objects and unlock all checked-out objects in one or more Project repositories that you select in the Clear Project Repository window. Perform the following steps:

1 Start SAS Data Integration Studio. A window displays with various options for maintaining a metadata profile.
2 Select a metadata profile that enables you to work as an unrestricted user. The default repository specified in this profile must be the parent repository on which the desired Project repositories depend (the Project repositories that you want to clear).
3 On the SAS Data Integration Studio desktop, select the `Inventory` tab.
4 In the Inventory tree, select the parent repository of the Project repository to be cleared, then select **Project ▶ Clear Project Repository** from the menu bar. The Clear Project Repository window displays. Unrestricted users see all project repositories that depend on the selected repository.
5 Has the Project repository been deleted? If not, skip to the next step. If so, select `Search for deleted project repository information`. Unrestricted users see all deleted project repositories that depend on the selected repository.
6 In the Clear Project Repository window, select one or more project repositories to be cleared. Then, click `OK`. All new objects are deleted. All checked-out objects are unlocked. All changes that have not been checked in are lost.

Note: An *unrestricted user* (a special user who is specified for a metadata server) can clear all Project repositories that depend on the default repository in his or her metadata profile. For more information about administrative profiles, see “Create a Profile for an Administrator, with Change Management” on page 53. For details about the unrestricted user, administrators should see the "How to Designate an Unrestricted, Administrative, or Trusted User" section in the *SAS Intelligence Platform: Security Administration Guide*. △

Additional Information

The help for SAS Data Integration Studio provides more details about change management. To display the relevant Help topics, perform the following steps to :

1 From the SAS Data Integration Studio menu bar, select **Help ▶ Contents**. The Help window displays.

2 In the left pane of the Help window, select **Task Overviews ▶ SAS Data Integration Studio Task Reference ▶ Using Change Management in SAS Data Integration Studio**.

Specifying Global Options in SAS Data Integration Studio

Problem

You want to set default options for SAS Data Integration Studio.

Solution

Specify the appropriate start option or specify an option in the global Options window.

Tasks

Specify Start Options

You can specify one or more options in the start command or in the `etlstudio.ini` file. See "Starting SAS Data Integration Studio" on page 50.

Use the Global Options Window

To display the global Options window from the SAS Data Integration Studio desktop, select **Tools ▶ Options** from the menu bar.

From the Options window, you can specify options such as the following:

- the default support for case, special characters, or both in DBMS names
- the default SAS Application Server
- the default display options for the Process Designer window
- options for the code that SAS Data Integration Studio generates for jobs and transformations; parallel processing options are included
- options for Data Quality transformations that are available in the Process Library tree
- options for the View Data window that is available for tables, external files, and selected transformations

CHAPTER

4

Importing, Exporting, and Copying Metadata

About Metadata Management

This chapter describes how to export, import, copy, and paste selected metadata objects in SAS Data Integration Studio. The metadata can be in SAS Open Metadata Architecture format or in other formats.

The online Help for SAS Data Integration Studio has additional information about the import and export of SAS metadata and other metadata. To display the relevant topics, from the desktop select **Help ▶ Contents ▶ Task Overviews ▶ SAS Data Integration Studio Task Reference ▶ Importing and Exporting Metadata**.

For a more comprehensive view of metadata management, administrators should see the metadata management chapters in the *SAS Intelligence Platform: System Administration Guide*.

Working with SAS Metadata

About Importing and Exporting SAS Metadata

SAS Data Integration Studio enables you to import and export metadata in SAS Open Metadata Architecture format. This feature enables you to reuse the metadata for tables, jobs, and other objects. For example, you can develop a job in a test environment, export it, then import the job into a production environment

From the desktop, you can select the object or objects to be exported and export them. The metadata is saved to a SAS Package file. You then import the package in SAS Data Integration Studio and save it to the same metadata repository or to a different metadata repository. The source and target repository can be located on the same host machine or on different host machines.

The source metadata server and the target metadata server can run on host machines that have different operating systems. For example, you can import objects into a metadata repository on a UNIX system even though those objects were exported from a Microsoft Windows system.

The metadata Export Wizard and Import Wizard enable you to perform the following tasks:

- export the metadata for one or more selected objects in the Inventory tree, Custom tree, or Project tree.

- export the metadata for all objects in one or more selected folders in the Custom tree.
- export access controls that are associated with exported objects (optional).
- export data, dependent metadata, and other content that is associated with exported objects (optional).
- change object names, port numbers, host machine names, schema names, physical paths, and other attributes when you import metadata (optional). For example, you can export the metadata for a SAS table, and upon import, change the metadata so that it specifies a DBMS table in the target environment.

SAS Metadata That Can Be Imported and Exported

You can import and export metadata for the following types of objects in SAS Data Integration Studio:

- libraries
- physical tables
- external files
- jobs
- generated transformations
- output from a Mining Results transformation
- stored processes (including jobs deployed for execution by a SAS stored process server or a Web service client)
- information maps
- notes
- documents
- folders containing the previous objects, including the folder for a repository

Note: The metadata Export Wizard is optimized for selective metadata export, not the export of whole repositories. The Export Wizard will not export the metadata for servers, database schemas, or users, for example. △

For more information about the export and import of whole metadata repositories, administrators should see the metadata management chapters in the *SAS Intelligence Platform: System Administration Guide*, especially the "Creating, Moving, Copying, and Deleting Metadata Repositories" chapter.

Automatic Content

When you select metadata objects for export, some content is automatically moved with them. The following table describes the content that is automatically moved with some objects.

Table 4.1 Automatic Content

Exported Metadata Object	Automatic Content
job	□ attributes (Name, Description, Roles, Extended Attributes, etc.) □ documents and notes □ parameters □ Automap setting for transformations □ Message Queue objects □ custom code associated with a job, a transformation, a pre-process, a post-process, or a custom action code (status-handling action code)
physical table	□ attributes (Name, Description, Roles, Extended Attributes [including Web Stream flags], etc.) □ documents and notes □ parameters □ primary keys and unique keys, but not foreign keys
external file	□ attributes (Name, Description, Roles, Extended Attributes [including Web Stream flags], etc.) □ documents and notes □ parameters □ primary keys and unique keys, but not foreign keys □ external format files □ COBOL copybooks
generated transformation	□ attributes (Name, Description, Roles, Extended Attributes (including Web Stream flags), etc.) □ documents and notes □ default settings for option values

Keep the following in mind about the content that is automatically moved with some objects:

- When importing a job, or when using the **`Paste Special`** option to paste a job, you can specify where any custom code for the job will be written on the target system.
- When importing an external file, or when using the **`Paste Special`** option to paste an external file, you can specify that the external file and any format files (external format files or COBOL copybooks) are written to separate locations on the target system.

Optional Content

When you select metadata objects for export, you have the option to move some of the content. Later, you can decide whether you want to import this optional content. The following table describes the optional content that can be moved with some objects.

Table 4.2 Optional Content

Exported Metadata Object	Optional Content
job	□ metadata for any libraries, tables, external files, notes, and documents in the job □ metadata for any generated transformations in the job □ metadata for the output of a Mining Results transformation in the job □ metadata for any job nested within the exported job
SAS library or a DBMS library	□ metadata for the tables in the library
SAS XML library	□ XML file that is specified in the library metadata □ any XML Map file that is specified in the library metadata
SAS data set	□ physical SAS data set □ SAS constraints: check, not null, primary keys and unique keys, but not foreign keys
DBMS table	□ physical DBMS table □ DBMS constraints: none
slowly changing dimension (SCD) table	□ physical SCD table □ any physical cross-reference table that is specified in the SCD table metadata
external file	□ physical external file

Keep the following in mind about the optional content that can be moved with some objects:

- Metadata for DBMS table constraints is not exported or imported.
- Avoid using the metadata Export and Import Wizards to move large data tables along with the metadata for those tables. All exported content must be packaged into a SAS package file, and it is not efficient to add large data tables to the package.
- Foreign keys, user-defined column formats and informats, and ODS tag sets associated with a SAS XML library are not handled by the metadata Export and Import Wizards and must be moved manually.
- The best way to export metadata for XML tables is to export the metadata for the parent SAS XML library.

Restoration of Metadata Associations

Some metadata objects depend on other metadata objects in order to work properly. For example, the metadata for a table must be associated with the metadata for a library, or the table will not be accessible in SAS Data Integration Studio.

If you import metadata objects that depend on other metadata, the Import Wizard will prompt you to associate the imported objects with appropriate metadata in the target environment. Depending on the object that you are importing, the Import Wizard might prompt you to restore an association between the imported object and the following metadata in the target environment:

- □ a SAS Application Server
- □ a WebDAV server
- □ base path on a WebDAV server
- □ libraries
- □ tables
- □ external files
- □ physical locations for external files or for libraries
- □ OLAP schemas
- □ output from a Mining Results transformation
- □ a source code repository and an archive path (for stored processes)

The Import Wizard enables you to select a default match, if any, between the imported metadata and metadata that exists in the target environment. The next table describes the criteria for a default match.

Table 4.3 Criteria for a Default Match

Exported Metadata Object	Matching Criteria
folder	Parent Folder.Metadata Name for the folder
job	Folder.Metadata Name for the job
library	SAS Application Server.Metadata Name for the library
physical table	Library.Metadata Name for the table
external file	Folder.Metadata Name for the file
generated transformation	Unique ID for the transformation (each generated transformation has a unique ID)

You can restore the same metadata associations that were specified in the source environment, or you can specify different associations. The Import Wizard also enables you to select **None** (no matching object in the target environment). You might want to select **None** if there is no appropriate match at the time of import, and you intend to supply the missing metadata association later.

Optional Promotion of Access Controls

You can include direct assignments of access control entries (ACEs) and access control templates (ACTs) in the promotion process. The following limitations apply to the promotion of access controls:

The wizards cannot promote ACTs, but they can promote direct associations between ACTs and objects. In order for an object's ACT association to be promoted, an ACT of the same name must exist in the destination metadata repository.

Access control entries and ACT associations that are inherited from a folder are not promoted. For inheritance to take effect in the destination metadata repository, you must also promote the folder that contains the direct access control entries or ACT association.

Access control entries are not promoted if they refer to users or groups that do not exist in the destination metadata repository (or in the parent Foundation repository, if the destination is a dependent repository).

If you do not specify that you want to include access controls in the promotion process, then access controls are applied as follows:

- If you import an object that already exists in the destination metadata repository, then the permissions that have been applied to the existing object are preserved when you overwrite the object.
- If you import an object for the first time, then the object inherits permissions from the folder into which the object is imported.

Logging after Import or Export

The metadata Import and Export Wizards create a log for the current session. The log can be viewed from the Finish panel of the wizards. The export log is included in the SAS package file upon export.

On Windows systems, metadata export and import logs are written to the user's folder at a location such as the following: `Documents and Settings\user name\My Documents`. The log file is named `Import_date/time.log` or `Export_datetime.log`.

Best Practices for Importing or Exporting SAS Metadata

If possible, keep sets of related jobs, tables, and other objects under the same parent folder in the Custom tree. Exporting a parent folder structure will provide a well-defined container for changes to be exported.

CAUTION:

If an import operation will result in the update of existing objects in the target repository, make sure that the update is necessary. △

Importing over an existing object, such as a table, overwrites dependent metadata objects with new IDs, which can impact the column mappings in existing jobs. If there are no changes being made to existing objects, you should not include them in the import process. The Import Wizard enables you to select only those objects that you want to import.

If you have to move a large volume of metadata from one repository to another, such as 100 jobs and their related tables, consider using full metadata repository promotion instead of the Export Wizard and Import Wizard in SAS Data Integration Studio. For more information about the export and import of whole metadata repositories, administrators should see the metadata management chapters in the *SAS Intelligence Platform: System Administration Guide*, especially the "Creating, Moving, Copying, and Deleting Metadata Repositories" chapter.

For more information about best practices, see the technical white paper, “Best Practices for SAS®9 Metadata Server Change Control,” at the following location: `http://support.sas.com/documentation/whitepaper/technical`.

Preparing to Import or Export SAS Metadata

Some metadata import and export features require the latest metadata server, as described in “Required Components for SAS Data Integration Studio” on page 48.

Metadata import tasks require planning and preparation of the target environment. Typical preparation tasks include the following:

- Verify that the appropriate servers are running in the source and target environments.

- □ Set up Custom tree folders in the source repository, so the objects can be exported from a convenient folder structure.
- □ Set up Custom tree folders in the target repository, so the objects can be imported into a convenient folder structure.
- □ Set up security for folders and objects in the target environment.
- □ For stored processes, set up a source code repository in an archive path (if needed) in the target environment.

For more information about preparing to use the metadata Import and Export Wizards, administrators should see the “Promoting Individual BI Objects” chapter in the *SAS Intelligence Platform: System Administration Guide*.

Exporting SAS Metadata

Problem

You want to export selected metadata objects from SAS Data Integration Studio so that you can import them later.

Solution

Use the Export Wizard to export the metadata. You can then import the package in SAS Data Integration Studio and save it to same metadata repository or to a different metadata repository. The source and target repository can be located on the same host machine or on different host machines.

Tasks

Document the Metadata That Will Be Exported (optional)

Metadata export and import tasks will be easier to manage if you create a document that describes the metadata to be exported, the metadata that should be imported, and the main metadata associations that must be reestablished in the target environment. Otherwise, you might have to guess about these issues when you are using the Export Wizard and the Import Wizard.

Export Selected Metadata

Perform the following steps:

1. Start SAS Data Integration Studio. A window displays that has various options for maintaining a metadata profile.
2. Select a profile that enables you to connect to the metadata repository with the objects to be exported.
3. In a tree view, right-click the selected objects and select `Export` from the pop-up menu. The Export wizard displays. Alternatively, you can left-click the objects to be exported and select **File ▶ Export** from the menu bar.

4 In the first page of the wizard, specify a path and name for the export package or accept the default. When finished, click **`Next`**.

5 Review the list of objects that you have selected for export. Deselect the check box for any objects that you do not want to export.

6 If desired, click an object, and then click the **`Options`** tab to view its options. For example, you can click the **`Options`** tab to specify whether or not you want to export content with the object. When you are finished, click **`Next`**.

7 Review the metadata to be exported. When finished, click **`Export`**. The metadata is exported to a SAS package file. A status page displays, indicating whether the export was successful. A log with a datetime stamp is saved to your user directory.

8 If desired, click **`View Log`** to view a log of the export operation. When you are finished, click **`Finish`**.

Importing SAS Metadata

Problem

You want to import metadata into SAS Data Integration Studio that was exported with the Export Wizard.

Solution

Use the Import Wizard to import the SAS package file that contains the metadata. The package can be saved to the same metadata repository or to a different metadata repository. The source and target repository can be located on the same host machine or on different host machines.

Tasks

Identify the Metadata That Should Be Imported (optional)

It will be easier to import metadata if you have a document that describes the metadata that was exported, the metadata that should be imported, and the main metadata associations that must be reestablished in the target environment.

For example, suppose that a SAS Data Integration Studio job was exported. When you import the job, the Import Wizard will prompt you to associate tables in the job with libraries in the target environment. If appropriate libraries do not exist, you might have to cancel the wizard, register appropriate libraries, and run the wizard again. However, if the library requirements are known and addressed ahead of time, you can simply import the tables and specify an appropriate library in the target environment.

Import the SAS Package File

Perform the following steps:

1 Start SAS Data Integration Studio. A window displays that has various options for maintaining a metadata profile.

2 Select a profile that enables you to connect to the metadata repository into which you will import the exported metadata.

3 In a tree view, right-click the icon for the target repository (or a folder in the target repository) and select **`Import`** from the pop-up menu. The Import wizard displays. Alternatively, you can left-click the icon for the target repository (or a folder in the target repository) and select **File ▶ Import** from the menu bar.

4 In the first page of the wizard, select the package to be imported. Select the option to import all objects in the package or just the new objects (objects which do not exist in the target repository). When finished, click **`Next`**.

5 Review the list of objects that you have selected for import. Deselect the check box for any objects that you do not want to import.

6 If desired, click an object, and then click the **`Options`** tab to view its options. For example, you can click the **`Options`** tab to specify whether or not you want to import content, if content was exported with the object. When finished, click **`Next`**.

7 Review the metadata associations to be restored. For example, if you are importing a table, you will prompted to specify a library for that table. Click **`Next`** to begin restoring the required associations.

8 You will be presented with one or more pages that prompt you to associate the imported object with other objects (such as libraries and servers). When finished, click **`Next`**.

9 Review the metadata to be imported. When finished, click **`Import`**. The metadata is imported. A status page displays, indicating whether the import was successful. A log with a datetime stamp is saved to your user directory.

10 If desired, click **`View Log`** to view a log of the import operation. When finished, click **`Finish`**.

Copying and Pasting SAS Metadata

Problem

You want to create a metadata object that is similar to another metadata object in a SAS Data Integration Studio tree view.

Solution

Use the **`Copy`** and **`Paste`** menu options to create a copy of the object, then modify the copy as desired. As an alternative to **`Paste`**, you can use **`Paste Special`** to display the Import Wizard, which enables you to select which attributes are copied and to change some attributes in the pasted object.

Tasks

Copy

To copy an object in a tree view, right-click the object and select **`Copy`** from the pop-up menu.

Paste

To paste a copy in which almost all of the attributes are the same as the original, right-click the icon for the target repository, then select `Paste` from the pop-up menu. If you are working under change management, the new object will appear in the Project tree. To prevent duplication, the new object will be named "Copy of object_name."

To paste a copy into a different Custom tree folder, right-click the target folder and select `Paste` from the pop-up menu. If you are working under change management, the new object will appear in the Project tree. The new object will have the same name as the original, but if you check the properties of the new object, you will see that it is associated with the folder that you selected. (Duplicate names in different Custom tree folders are valid.)

Paste Special

After using the `Copy` option on a selected object, you can display a wizard that will enable you to select what attributes are copied and to change some attributes in the pasted object. Right-click a target folder in the Custom tree, then select `Paste Special` from the pop-up menu. The Import Wizard will be displayed. The general steps for using this wizard are described in "Importing SAS Metadata" on page 73.

Working with Other Metadata

About Exporting and Importing Other Metadata

SAS Data Integration Studio supports the SAS Open Metadata Architecture, but you can import metadata in other formats, and you can export SAS metadata to other formats. Supported metadata formats include Common Warehouse Metamodel (CWM) format or a format that is supported by a SAS Metadata Bridge. For example, a modeling application can be used to create a data model that specifies a set of tables, columns, indexes, and keys. The model can be exported in CWM format, and you can import it into SAS Data Integration Studio.

The Metadata Importer option enables you to perform the following tasks:

- Import relational metadata in CWM format or a format that is supported by a SAS Metadata Bridge.
- Perform change analysis on the imported metadata (compare imported metadata to existing metadata).
- View any changes in the Differences window.
- Run impact analysis or reverse impact analysis on tables and columns in the Differences window, to help you understand the impact of a given change on the target environment.
- Choose which changes to apply to the target environment.

The Metadata Exporter option enables you to export metadata in the default metadata repository that is specified in your metadata profile, in CWM format, or a format that is supported by a SAS Metadata Bridge.

Other Metadata That Can Be Imported and Exported

You can import and export relational metadata in CWM format or in a format accessible with a SAS Metadata Bridge. Relational metadata includes the metadata for the following objects:

- tables
- columns
- indexes
- keys (including primary keys and foreign keys)

Usage Notes for Importing or Exporting Other Metadata

- Avoid importing the same DBMS data model as new multiple times. A DBMS data model specifies a library and schema for each table in the model. If you use the Metadata Import wizard to import the same DBMS data model multiple times into the same metadata repository, and you use the Import as new metadata option each time, you will create duplicate metadata for schemas and libraries in your repository. Duplicate schema names can cause unpredictable results when you attempt to compare the newest, imported metadata from the model with existing metadata.
- You cannot run change analysis on metadata that is imported from z/OS systems.
- If you are working under change management, it is a good practice to check in the comparison result metadata before viewing or applying the results.
- When imported metadata is compared to existing metadata, the differences between the two are stored in a comparison result library. In the current release, the comparison result library cannot be a SAS/SHARE library. Accordingly, in an environment where two or more people will be performing change analysis on imported metadata, care should be taken to avoid contention over the same comparison results library. For example, each user can create his or her own comparison result library.
- To avoid problems that arise when character sets from different locales are combined in the same comparison result library, create one or more comparison result libraries for each locale.
- SAS uses the combination of a DBMS server, a DBMS library, and a schema to access a DBMS table. When you select a DBMS server in the Metadata Import wizard, it is assumed that the metadata for the corresponding library and schema exist in a current metadata repository. If they do not exist, you will receive an error when you try to import the metadata, indicating that a required schema is missing.
- If you are working under change management, empty your Project tree of any metadata before importing more metadata with the Metadata Importer. This will make it easier to manage the imported metadata from a particular session. If you want to save any metadata in the Project tree, check in the repository. Then select **Project ▶ Clear Repository**.
- The Metadata Import wizard enables you to select a metadata file that is local or remote to SAS Data Integration Studio. Remote support is provided for Windows and UNIX hosts only.
- Under change management, imported metadata is compared to checked-in metadata. Accordingly, any metadata in the Project repository that has not been checked in will not be included in the comparison.

If you mistakenly run a comparison before the appropriate metadata has been checked in, you can check in the contents of the Project repository and then select **`Comparison Recompare`** from the menu bar.

- Null SAS formats that show as differences in change analysis will, when applied, overwrite user-defined SAS Formats in a metadata repository. Be careful when you apply formats during change analysis.

Preparing to Import or Export Other Metadata

In SAS Data Integration Studio , no preparation is required in order to import or export metadata in CWM format. To import or export metadata in a format that is accessible with a SAS Metadata Bridge, you must license the appropriate SAS Metadata Bridge. For more information, contact your SAS representative.

Importing As New Metadata (without Change Analysis)

Problem

You want to import a data model for a set of new tables. The model is in CWM format or a format that is accessible with a SAS Metadata Bridge. You are certain that none of the imported metadata is a duplicate of existing metadata.

Solution

Use the **`Import as new metadata`** option in the Metadata Import Wizard.

The **`Import as new metadata`** option specifies that metadata in the selected file will be imported without comparing it to existing metadata. This eliminates some steps, but it also means that you can import duplicate metadata. Under change management, the imported metadata appears in your Project tree, where you can review it before checking it in. Without change management, all metadata in the selected file will be registered to the default repository in the current profile.

Tasks

Import As New Metadata (No Change Analysis)

Perform these steps to use the **`Import as new metadata`** option in the Metadata Import Wizard.

1. From the desktop, select **Tools ▶ Metadata Importer** from the menu bar. The Metadata Import Wizard displays.
2. From the Metadata Import wizard, select the format of the file that you want to import, such as CWM Import. Click **`Next`**.
3. From the File Location window, specify a path to the file that contains the metadata to be imported, such as OrionStar.xml. (The path must be accessible to the default SAS Application Server for SAS Data Integration Studio or to another SAS Application Server that you select, using the **`Advanced`** button.) Click **`Next`**.

4 From the Import Selection window, select **`Import as new metadata`** and click **`Next`**.

5 From the Metadata Location window, specify a single SAS library or DBMS server for all of the table metadata that is being imported from the selected file. In this case, select Library A. Click **`Next`**.

6 In the Folder Selection window, select a Custom tree folder for any new table metadata that is being imported. (Updated table metadata will keep any current Custom tree folder that is specified for it.)

7 In the Finish window, review the metadata and click **`Finish`** when you are finished. All tables that are specified in the imported metadata are registered to the default repository. The imported tables might not appear in the Project tree until you refresh the tree.

8 Right-click the Project tree and select **`Refresh`** from the pop-up menu. The metadata for the imported tables appears in the Project tree.

Importing Metadata with Change Analysis

Problem

You want to you want to import a data model for a set of tables. The model is in CWM format or a format that is accessible with a SAS Metadata Bridge. It is possible that some of the imported metadata is a duplicate of existing metadata.

Solution

Use the **`Compare import metadata to repository`** option in the Metadata Import wizard.

The **`Compare import metadata to repository`** option specifies that metadata in the selected file will be imported and compared to existing metadata. Differences in tables, columns, indexes, and keys are detected. Under change management, imported metadata is compared to checked-in metadata that is associated with the library or DBMS server that you selected in the wizard. Without change management, imported metadata is compared to the metadata in the default repository that is associated with the selected library or DBMS server. Differences will be stored in a comparison result library. You can view the changes in the Differences window.

Tasks

Compare Import Metadata to Repository

Perform these steps to use the **`Compare import metadata to repository`** option in the Metadata Import Wizard.

1 From the SAS Data Integration Studio desktop, select **Tools ▶ Metadata Importer**. The Metadata Import wizard displays.

2 From the Metadata Import wizard, select the format of the file that you want to import, such as **`CWM Import`**. Click **`Next`**.

3 From the File Location window, specify a path to the file that contains the metadata to be imported. (The path must be accessible to the default SAS Application Server for SAS Data Integration Studio or to another SAS Application Server that you select, using the **`Advanced`** button.) Click **`Next`**.

4 From the Import Selection window, select **`Compare import metadata to repository`**. The **`Comparison result library`** field becomes active.

5 In the **`Comparison result library`** field, select a comparison result library. You can change the default options for the comparison by clicking the **`Advanced`** button to display the Advanced Comparison Options window. Click **`Next`**.

6 From the Metadata Location window, specify a SAS library or DBMS server for all of the table metadata that is being imported from the selected file. Click **`Next`**.

7 In the Folder Selection window, select a Custom tree folder for any new table metadata that is being imported. (Updated table metadata will keep any current Custom tree folder that is specified for it.) Click **`Next`**.

8 In the Finish window, review the metadata and click **`Finish`** when you are finished. The imported metadata will be compared to checked-in metadata. If the comparison is successful, a dialog box displays, indicating that the comparison has been completed.

9 You are given the option of viewing a log of the comparison process or closing the dialog box. The Comparison Results tree displays a comparison results object that is named after the imported file.

10 If you are working under change management, it is a good practice to check in the comparison result metadata before viewing or applying the results. From the Project tree, right-click the Project repository icon and select **`Check In Repository`**.

View the Comparison

Perform the following steps to view the results of an import metadata comparison.

1 In the Comparison Result tree, select the comparison results object. The comparison results object is named after the imported file, and it has an XML extension. The Comparison menu displays on the menu bar.

2 From the menu bar, select **Comparison ▶ View Differences**. The Differences window displays.

The Differences window is divided into two panes: Import Metadata and Repository Metadata. The Import Metadata pane displays metadata that is being imported. Under change management, the Repository Metadata pane displays any matching metadata in the change-managed repository. Only checked-in metadata displays. Without change management, the Repository Metadata pane displays any matching metadata in the default repository.

Any items that are new, changed, or deleted in the imported metadata are indicated with icons and displayed in a special background color.

Select and Apply Changes

Perform the following steps to select certain imported metadata and apply it to the current metadata repository.

1 Use the check boxes in the Import Metadata pane to select which tables and columns to import.

2 Click **`Apply`**. The Apply Confirmation window displays.

3 Click **`OK`** to accept the changes. When the new metadata has been registered to the default repository, the Difference window will be empty.

Applying Changes to Tables with Foreign Keys

When you import metadata about a set of tables that are related by primary keys or foreign keys, and the keys have been either added or updated in the imported metadata, do one of the following:

- apply all changes in the imported metadata, or
- apply selective changes, making sure to select all tables that are related by primary keys or foreign keys

Otherwise, the key relationships will not be preserved.

Restoring Metadata for Foreign Keys

When you apply changes from imported metadata, a warning message is displayed if foreign key metadata is about to be lost. At that time, you can cancel or continue the apply operation. However, if you accidentally lose foreign key metadata as a result of an apply operation, it is possible to restore this metadata.

Assuming that the imported metadata correctly specifies the primary keys or foreign keys for a set of tables, you can recompare the imported metadata to the metadata in the repository. In the Comparison Results window, select the icon for the appropriate comparison result and then select **`Comparison Recompare`** from the menu bar. In the Differences window, accept all changes, or select the primary key table and all related foreign key tables together and apply changes to them.

After you import the metadata for a table, you can view the metadata for any keys by displaying the properties window for the table and clicking the **`Keys`** tab.

Deleting an Invalid Change Analysis Result

When you perform change analysis on imported metadata, it is possible to import the wrong metadata or compare the imported metadata to the wrong current metadata. If this happens, the comparison result metadata in the Comparison Result tree will not be valid, as well as the data sets for this comparison in the comparison result library.

If you are not working under change management, delete the bad comparison result metadata.

If you are working under change management, perform the following steps:

1. Check in the bad comparison result metadata. From the Project tree, right-click the **`Project`** repository icon and select **`Check In Repository`**. This will make the comparison result metadata available to others, such as the administrator in the next step.
2. In SAS Data Integration Studio, have an administrator open the repository that contains the bad comparison result metadata.
3. Have the administrator delete the bad comparison result from the Comparison Results tree. This will delete both the metadata and the data sets for a comparison result.

Exporting Other Metadata

Problem

You want to export metadata from SAS Data Integration Studio in CWM format or a format that is supported by a SAS Metadata Bridge. Some SAS solutions rely on this method.

Solution

Use the Metadata Export wizard to export the default metadata repository that is specified in your metadata profile.

Tasks

Export the Default Metadata Repository

Perform the following steps to export the default metadata repository:

1. From the SAS Data Integration Studio desktop, select **Tools ▶ Metadata Exporter**. The Metadata Export wizard displays.
2. Select the format for the exported repository and click **`Next`**. If the selected file format requires you to specify options, an options window displays.
3. If you must specify options for the export format, specify these and click **`Next`**.
4. Specify a path for the export file and click **`Next`**.
5. Review the metadata in the Finish window and click **`Finish`** when you are finished.

CHAPTER
5

Working with Tables

About Tables

A table contains data arranged in rows and columns. SAS Data Integration Studio supports SAS tables and the tables created by the database management systems supported by SAS Access software. The main places tables are displayed within SAS Data Integration Studio include:

- the Inventory, Custom, and Project trees
- the Process Designer window

The following table shows some common tasks that you perform when you work with tables in SAS Data Integration Studio:

Table 5.1 Common Table Tasks

If you want to	Do this
Work with a table in SAS Data Integration Studio.	Register the table. See the sections about source designers and the Target Table wizard in this chapter for more information. See also "Registering Sources and Targets" on page 56.
Specify a registered table as a source or a target in a SAS Data Integration Studio job.	Select the table in a tree. Then, drag it to the Process Designer window for the job and drop it onto the appropriate drop zone. You can also select target tables when you create a job in the New Job wizard.
View the data or metadata for a registered table.	Use the View Data window to review table data and metadata in the trees and the Process Designer window. See "Browsing Table Data" on page 109 for more information.
Import, export, copy, and paste table metadata.	Use the Metadata Importer and Metadata Exporter wizards. See "Working with SAS Metadata" on page 66.

Registering Tables with a Source Designer

Problem

You want to create a job that includes one or more tables that exist in physical storage, but the tables are not registered in a metadata repository.

Solution

Use the appropriate source designer to register the tables. Later, you can drag and drop this metadata into the target position in a process flow. When the process flow is executed, SAS Data Integration Studio will use the metadata for the target table to create a physical instance of that table.

The first page of the wizard will prompt you to select a library that contains the tables to be registered. (Typically, this library has been registered ahead of time.) SAS

Data Integration Studio must be able to access this library. For details about SAS libraries, see "Libraries" on page 8.

Tasks

Register a Table with a Source Designer

Perform the following steps to register one or more tables that exist in physical storage:

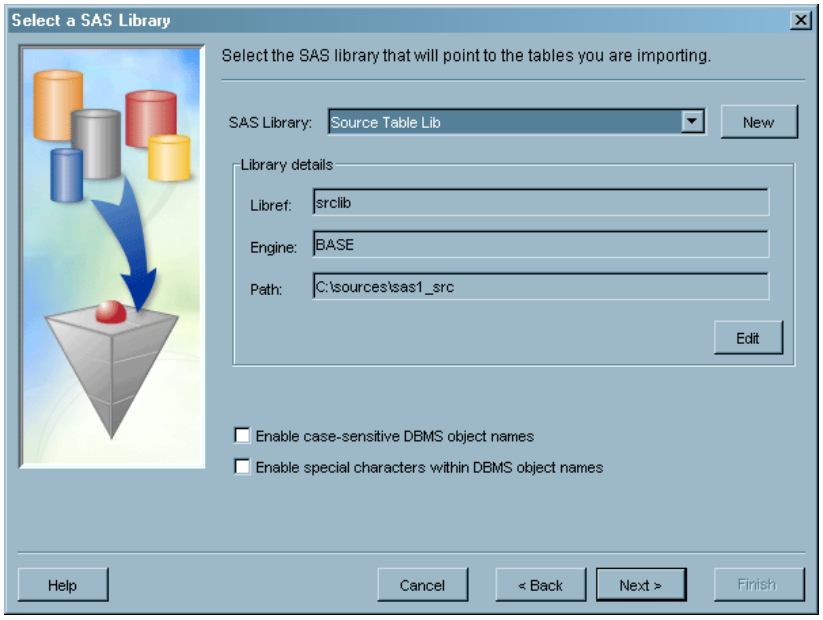
1 Display the source designer selection window in one of the following ways:
   - ☐ Click `Source Designer` on the shortcuts bar.
   - ☐ Select **Tools ▶ Source Designer**.
   - ☐ Select **File ▶ New Object**. Then, click `Source Table` on the New Object Wizard window.

2 When the source designer selection window opens, only those data formats for which source designers have been installed are available. Select the appropriate wizard for the data type of the tables that you need to register.

3 Click `Next`. The wizard tries to open a connection to the default SAS Application Server. If there is a valid connection to this server, you might be prompted for a user name and a password. After you have provided that information, you will be taken directly to the Select a SAS Library window.

4 Select the SAS library that contains the tables that you want to register, and review the settings displayed in the `Library Details` section of the window. Sample settings are shown in the following display.

Display 5.1 Sample Library Settings


Select a SAS Library

Select the SAS library that will point to the tables you are importing.

SAS Library: Source Table Lib

Library details

Libref: srclib

Engine: BASE

Path: C:\sources\sas1_src

☐ Enable case-sensitive DBMS object names

☐ Enable special characters within DBMS object names

5 Click `Next` to access the Define Tables window. Select one or more tables to register.

6 Click `Next` to access the Select Folder window. Specify a Custom tree group for the table or tables that you are registering. A Custom tree group is a folder that you

can use to keep similar types of metadata together in the Custom tree on the SAS Data Integration Studio desktop.

7 Click `Next` to access the Wizard Finish window. Review the metadata that will be created. When you are satisfied that the metadata is correct, click `Finish` to save the data and close the wizard.

Registering Tables with the Target Table Wizard

Problem

You want to create a job that includes a table that does not yet exist. This new table could hold the final results of the job, or it could serve as the input to a transformation that would continue the job.

Solution

Use The Target Table wizard to register the new table. Later, you can drag and drop this metadata onto the target position in a process flow. When the process flow is executed, SAS Data Integration Studio will use the metadata for the target table to create a physical instance of that table.

The physical storage page of the wizard will prompt you to select a library that will contain the table to be registered. (Typically, this library has been registered ahead of time.) For details about SAS libraries, see “Libraries” on page 8.

Tasks

Register a Table with a Target Table Wizard

Perform the following steps to register a table that does not exist:

1 Display a selection window in one of the following ways:
   - □ Click `Target Designer` on the shortcuts bar.
   - □ Select **Tools ▶ Target Designer**.
   - □ Select **File ▶ New Object**. Then, click `Target Table` on the New Object Wizard window.

2 The target designer selection screen opens. Select `Target Table`. Note that the list of wizards includes only target types for the Target Designer wizards that have been installed.

3 Click `Next`. Because you have set up a default SAS Application Server, the wizard tries to open a connection to this default server. If a valid connection to this server exists, you are prompted for a valid user name and a password. After you provide that information, the name and description window displays in the Target Table Designer. Enter a name and description for the table that you want to register. Note that the metadata object might or might not have the same name as the corresponding physical table. You will specify a name for the physical table in a later window in this wizard.

4 Click **Next** to access the Table Storage Information window. Enter appropriate values in the following fields:

- ☐ **DBMS**
- ☐ **Library**
- ☐ **Name** (must follow the rules for table names in the format that you select in the **DBMS** field. For example, if SAS is the selected DBMS, the name must follow the rules for SAS data sets. If you select another DBMS, the name must follow the rules for tables in that DBMS. For a SAS table or a table in a database management system, you can enable the use of mixed-case names or special characters in names.)
- ☐ **Schema** (if required by DBMS type)

Use the Table Storage Information window to specify the format and location of the table that you are registering. You also specify the database management system that is used to create the target, the library where the target is to be stored, and a valid name for the target. You can specify new libraries or edit the metadata definitions of existing libraries using the **New** and **Edit** buttons. You can use the **Table Options** button to specify options for SAS tables and tables in a DBMS. The following display shows these settings for a sample table.

Display 5.2 Sample Table Storage Settings

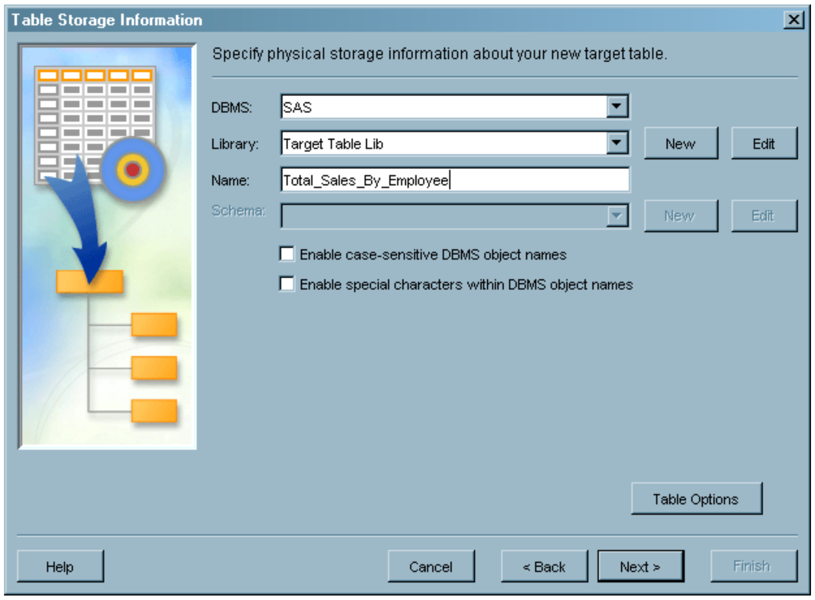


Table Storage Information
Specify physical storage information about your new target table.
New
Edit
New
Edit
Table Options
Help
Cancel
< Back
Next >
Finish

DBMS: SAS
Library: Target Table Lib
Name: Total_Sales_By_Employee
Schema:
☐ Enable case-sensitive DBMS object names
☐ Enable special characters within DBMS object names

5 Click **Next** to access the Select Columns window. Use the Select Columns window to import column metadata from existing tables registered for use in SAS Data Integration Studio. Drill down in the **Available Columns** field to find the columns that you need for the target table. Then, move the selected columns to the **Selected Columns** field.

6 Click **Next** to access the Change Columns/Indexes window. Use this window to accept or modify any column metadata that you selected in the Select Columns window. You can add new columns or modify existing columns in various ways. (For details, click the **Help** button for the window.)

7 Click **Next** when you are finished reviewing and modifying the column metadata. If you change the default order of the column metadata, you are prompted to save the new order. The Select Folder window is displayed. Specify a Custom tree

group for the table or tables that you are registering. A Custom tree group is a folder that you can use to keep similar types of metadata together in the Custom tree on the SAS Data Integration Studio desktop.

8 Click **Next** to access the Wizard Finish window. Review the metadata that will be created. When you are satisfied that the metadata is correct, click **Finish** to save the data and close the source designer wizard.

Viewing or Updating Table Metadata

Problem

You want to view or update the metadata for a table that you have registered in SAS Data Integration Studio.

Solution

You can access the properties window for the table and change the settings on the appropriate tab of the window. The following tabs are available on properties windows for tables:

- General
- Columns
- Indexes
- Keys
- Physical Storage
- Parameters
- Notes
- Extended Attributes
- Advanced

Use the properties window for a table to view or update the metadata for its columns, keys, indexes, and other attributes. You can right-click a table in any of the trees on the SAS Data Integration Studio desktop or in the Process Designer window. Then, click **Properties** to access its properties window.

Note that any updates that you make to a table change the physical table when you run a job that contains the table. These changes can have the following consequences for any jobs that use the table:

- Changes, additions, or deletions to column metadata are reflected in all of the jobs that include the table.
- Changes to column metadata often affect mappings. Therefore, you might need to remap your columns.
- Changes to keys, indexes, physical storage options, and parameters affect the physical external file and are reflected in any job that the includes the table.

You can use the impact analysis and reverse impact tools in SAS Data Integration Studio to estimate the impact of these updates on your existing jobs. For information see “About Impact Analysis and Reverse Impact Analysis” on page 255.

Using a Physical Table to Update Table Metadata

Problem

You want to ensure that the metadata for a table matches the columns contained in the physical table. If the metadata does not match the columns in the physical table, you need to update the metadata to match the physical table.

Solution

You can use the update table metadata feature. This feature compares the columns in a physical table to the columns that are defined in the metadata for that table. If the column metadata does not match the columns in the physical table, the metadata is updated to match the physical table.

For existing tables, the update table metadata feature adds new columns, removes deleted columns, and records changes to all of the column attributes. When you select and run this feature against one or more tables simultaneously, the update log lists which tables have been successfully updated and which have failed. When you use the update table metadata option on a physical table in DBMS format and the DBMS table has more than one schema, the update table metadata option selects the first schema.

The update table metadata feature uses the following resources:

- the current metadata server and the SAS Application Server to read the physical table
- the current metadata server to update the metadata to match the physical table

A warning message displays if the SAS Workspace Server component of the SAS Application Server is older than SAS 9.1.3, Service Pack 3. See the usage note, "Some New Features Require the Latest Servers" in the "SAS Data Integration Studio Usage Notes" topic in SAS Data Integration Studio help. For release-specific information on this feature, see the "Update Table Metadata Cannot Be Used for Some Tables" usage note.

Tasks

Run Update Table Metadata

Perform the following steps to run the update table metadata feature:

1. Select one or more tables from a SAS Data Integration Studio tree. Then, click **Update Table Metadata** on the **Tools** menu. You might be prompted to supply a user name and password for the relevant servers.
2. When the update is finished, you can choose to view the resulting SAS log.

Specifying Options for Tables

Problem

You want to set options for tables used in SAS Data Integration Studio jobs, such as DBMS name options; library, name, and schema options; and compression scheme and password protection options.

Solution

You can set global options for tables on the **`General`** tab of the **`Options`** menu. The **`Options`** menu is available on the **`Tools`** menu on the SAS Data Integration Studio menu bar. You can set local options on the tabs available on the properties window for each table.

Tasks

Set Global Options for Tables

Table 5.2 Global Table Options

Option name	Description
Enable case-sensitive DBMS object names	Specifies whether the source designer and target designer will generate code that supports case-sensitive table and column names by default. If you do not select the check box, no case-sensitive support is provided. If you select the check box, support is provided.
Enable special characters within DBMS object names	Specifies whether the source designer and target designer will generate code that supports special characters in table and names by default. If you select the check box, support is provided by default. When you select this check box, the **`Enable case-sensitive DBMS object names`** check box is also automatically selected.

The global settings apply to any new table metadata object, unless the settings are overridden by a local setting. See “Supporting Case and Special Characters in Table and Column Names” on page 93 for more information about DBMS object names.

Set Local Options for Tables

You can set local options that apply to individual tables. These local options override global options for the selected table, but they do not affect any other tables. For example, you can set local DBMS name options on the **`Physical Storage`** tab of the properties window for a table. These DBMS name options are listed in the following table.

Table 5.3 Local Table Options on the Physical Storage Tab

Option name	Description
DBMS	Specifies the database management system (DBMS) where the table is stored. To select a DBMS from a list, click the down arrow. DBMSs that are valid in the current context are listed.
Library	Specifies a library that you can use to access the table. You can choose from a list of libraries that are in the Project repository or its parent repositories. To select a library, click the selection arrow. To create a new library, click **`New`**, which opens the New Library wizard. To edit the properties of the existing library, click **`Edit`**, which opens the properties window for the data library.
Name	Specifies the name of the table. The name must follow the rules for table names in the DBMS that is selected in the **`DBMS`** field. For example, if SAS is the selected DBMS, the name must follow the rules for SAS data sets. If you select another DBMS, the name must follow the rules for tables in that DBMS. If the table is used for iterative or parallel processing, the table name must be preceded by an ampersand (&) character (for example, &sourcetable). This usage ensures that the parameters set for the iteration are recognized and that the table is included when the iterative process works through the list of tables contained in the control table.
Schema	When present, this field specifies the name of the schema that is associated with the DBMS table that is specified in the **`Name`** field. A schema is a map or model of the overall data structure of a database. To select a schema that has been defined in a current metadata repository, click the selection arrow. To specify a new schema, click **`New`**, which opens the New Database Schema wizard. To edit the properties of the existing schema, click **`Edit`**, which opens the properties window for the schema. When adding or editing a schema, note that the name of the schema in the metadata must match the name of the schema in the DBMS exactly (including case).
Enable case-sensitive DBMS object names	When present, specifies whether SAS Data Integration Studio will generate code that supports case-sensitive table and column names for the current table. If the check box is deselected, case sensitivity is not supported. If the check box is selected, case sensitivity is supported. This option overrides the global option with the same name.
Enable special characters within DBMS object names	When present, specifies whether the source designers and the Target Table wizard will generate code that supports special characters in table and names by default. If you select the check box, support is provided by default. When you select this check box, the **`Enable case-sensitive DBMS object names`** check box is also automatically selected. This option overrides the global option with the same name.
Table Options	When present, displays the Table Options window, where you can specify a compression scheme, password protection, or other options for the current table.

See “Supporting Case and Special Characters in Table and Column Names” on page 93 for more information about DBMS object names.

You can set additional table options in the Table Options window. To access this window, click **`Table Options`** on the **`Physical Storage`** tab of the properties window for a table. These options are covered in following table.

Table 5.4 Local Table Options in the Table Options Window

Option name	Description
Compressed	Specifies the kind of compression used, if any, for a SAS data set. (You cannot compress SAS data views because they contain no data.) Compression reduces storage requirements, but it increases the amount of CPU time that is required to process the file. Although compression affects performance by increasing the amount of CPU time that is required to process the file, it significantly reduces storage requirements. You should therefore use the default value only if you have sufficient disk space. In particular, you will want to enable compression for files such as Web logs that have columns with large widths that are sparsely filled. Select one of the following: □ **`NO`** (default): The SAS data set will not be compressed. □ **`YES`**: The SAS data set will be compressed using Run Length Encoding (RLE). Use this method for character data. □ **`BINARY`**: The SAS data set will be compressed using Ross Data Compression (RDC). Use this method for medium to large (several hundred bytes or larger) blocks of binary data (numeric variables).
Encrypted	Specifies whether a SAS data set is encrypted (YES) or not encrypted (NO). You cannot encrypt SAS data views because they contain no data.
Additional Options	Specifies options for SAS data sets or views. Separate each option with a blank. The field is restricted to a maximum of 200 characters. You can specify a password for the table in this field. Use table option syntax, such as **`read=readpw write=writepw`**.

Supporting Case and Special Characters in Table and Column Names

About Case and Special Characters in SAS Names

Rules for SAS Names

By default, the names for SAS tables and columns must follow these rules:

- □ Blanks cannot appear in SAS names.
- □ The first character must be a letter (such as A through Z) or an underscore (_).
- □ Subsequent characters can be letters, numeric digits (such as 0 through 9), or underscores.
- □ You can use uppercase or lowercase letters. SAS processes names as uppercase, regardless of how you enter them.

- Special characters are not allowed, except for the underscore. Only in filerefs can you use the dollar sign ($), pound sign (#), and at sign (@)

The following SAS language elements have a maximum length of eight characters:

- librefs and filerefs
- SAS engine names and passwords
- names of SAS/ACCESS access descriptors and view descriptors (to maintain compatibility with SAS Version 6 names)
- variable names in SAS/ACCESS access descriptors and view descriptors

Beginning in SAS 7 software, SAS naming conventions have been enhanced to allow longer names for SAS data sets and SAS variables. The conventions also allow case-sensitive or mixed case names for SAS data sets and variables.

The following SAS language elements can now be up to 32 characters in length:

- members of SAS libraries, including SAS data sets, data views, catalogs, catalog entries, and indexes
- variables in a SAS data set macros and macro variables

For a complete description of the rules for SAS names, see the topic, "Names in the SAS Language" in *SAS Language Reference: Concepts*.

Case and Special Characters in SAS Names

By default, the names for SAS tables and columns must follow the rules for SAS names. However, SAS Data Integration Studio supports case-sensitive names for tables, columns, and special characters in column names if you specify the appropriate DBMS name options in the metadata for the SAS table, as described in “Enabling Name Options for Existing Tables” on page 97 or “Setting Default Name Options for Tables and Columns” on page 97. Double-byte character set (DBCS) column names are supported in this way, for example.

The DBMS name options apply to most tables in SAS format. The following are exceptions:

- Special characters are not supported in SAS table names.
- Leading blanks are not supported for SAS column names and are therefore stripped out.
- Neither the External File source designers nor SAS/SHARE libraries and tables support case-sensitive names for SAS tables or special characters in column names. When you use these components, the names for SAS tables and columns must follow the standard rules for SAS names.

About Case and Special Characters in DBMS Names

Overview

You can access tables in a database management system (DBMS), such as Oracle or DB2, through a special SAS library that is called a database library. SAS Data Integration Studio cannot access a DBMS table with case-sensitive names or with special characters in names unless the appropriate DBMS name options are specified in both of these places:

- □ in the metadata for the database library that is used to access the table, as described in "Enabling Name Options for a New Database Library" on page 95 or "Enabling Name Options for an Existing Database Library" on page 96
- □ in the metadata for the table itself, as described in "Enabling Name Options for a New Database Library" on page 95 or "Enabling Name Options for an Existing Database Library" on page 96

Use the following methods to avoid or fix problems with case-sensitive names or with special characters in names in DBMS tables.

DBMSs for Which Case and Special Characters are Supported

SAS Data Integration Studio generates SAS/ACCESS LIBNAME statements to access tables and columns that are stored in DBMSs. The following SAS/ACCESS LIBNAME statements have options that preserve case-sensitive names and names with special characters:

- □ DB2 z/OS
- □ DB2 UNIX/PC
- □ Informix
- □ MySQL
- □ ODBC
- □ OLE DB
- □ Oracle
- □ Microsoft SQL Server
- □ Teradata

Verify Name Options for Database Libraries

Perform the following steps to verify that the appropriate DBMS name options have been set for all database libraries that are listed in the Inventory tree.

1. From the SAS Data Integration Studio desk top, select the Inventory tree.
2. In the Inventory tree, open the **`Libraries`** folder.
3. Right-click a database library and select **`Display Libname`** from the pop-up menu. A SAS LIBNAME statement is generated for the selected library. In the LIBNAME statement, set both the **`Preserve DBMS table names`** option to **`YES`** and the **`Preserve column names as in the DBMS`** option.
4. If these options are not set correctly, update the metadata for the library, as described in "Enabling Name Options for an Existing Database Library" on page 96 .

Enabling Name Options for a New Database Library

The following steps describe how to specify name options for a new database library so that table and column names are supported as they would in the DBMS. This task is typically done by an administrator. It is assumed that the appropriate database server has been installed and registered, and the appropriate database schema has been registered. For more information about database servers and schemas, see the chapters about common data sources in the *SAS Intelligence Platform: Data Administration Guide*.

1 Follow the general instructions in "Registering Any Libraries That You Need" on page 55. In the first window of the New Library wizard, select the appropriate kind of database library and click **`Next`**.
2 Enter a name for the library and click **`Next`**.
3 Enter a SAS LIBNAME for the library, then click **`Advanced Options`**. The Advanced Options window displays.
4 In the Advanced Options window, click the **`Output`** tab. In the **`Preserve column names as in the DBMS`** field, select **`Yes`**.
5 Click **`OK`** and enter the rest of the metadata as prompted by the wizard.

Enabling Name Options for an Existing Database Library

Perform the following steps to update the existing metadata for a database library to support table and column names as they exist in the DBMS.

1 In SAS Data Integration Studio, click the **`Inventory`** tab to display the Inventory tree.
2 In the Inventory tree, expand the folders until the **`Libraries`** folder is displayed.
3 Select the **`Libraries`** folder and then select the library for which metadata must be updated.
4 Select **File ▶ Properties** from the menu bar. The properties window for the library displays.
5 In the properties window, click the **`Options`** tab.
6 On the **`Options`** tab, click **`Advanced Options`**. The Advanced Options window displays.
7 In the Advanced Options window, click the **`Output`** tab. In the **`Preserve column names as in the DBMS`** field, select **`Yes`**.
8 In the Advanced Options window, click the **`Input/Output`** tab. In the **`Preserve DBMS table names`** field, select **`Yes`**.
9 Click **`OK`** twice to save your changes.

Verify DBMS Name Options in Table Metadata

Perform the following steps to verify that the appropriate DBMS name options have been set for DBMS tables that are used in SAS Data Integration Studio jobs.

1 From the SAS Data Integration Studio desk top, select the Inventory tree.
2 In the Inventory tree, open the **`Jobs`** folder.
3 Right-click a job that contains DBMS tables and select **`View Job`** from the pop-up menu. The job opens in the Process Designer window.
4 In the process flow diagram for the job, right-click a DBMS table and select **`Properties`** from the pop-up menu.

5 In the properties window, click the **`Physical Storage`** tab.

6 Verify that the **`Enable case-sensitive DBMS object names`** option and the **`Enable special characters within DBMS object names`** option are selected.

7 If these options are not set correctly, update the metadata for the table, as described in “Enabling Name Options for Existing Tables” on page 97 .

Enabling Name Options for Existing Tables

Perform the following steps to enable name options for tables for which metadata has been saved to a metadata repository. These steps apply to tables in SAS format or in DBMS format.

1 From the SAS Data Integration Studio desktop, display the Inventory tree or another tree view.

2 Open the **`Tables`** folder.

3 Select the desired table and then select **File ▶ Properties** from the menu bar. The properties window for the table displays.

4 In the properties window, click the **`Physical Storage`** tab.

5 On the **`Physical Storage`** tab, select the check box to enable the appropriate name option for the current table. Select **`Enable case-sensitive DBMS object names`** to support case-sensitive table and column names. Select **`Enable special characters within DBMS object names`** to support special characters in table and column names.

6 Click **`OK`** to save your changes.

Set DBMS Name Options in the Source Designers

The first window in a DBMS source designer wizard enables you to select the library that contains the DBMS table or tables for which you want to generate metadata. In the first window, check the boxes labeled **`Enable case-sensitive DBMS object names`** and **`Enable special characters within DBMS object names`**.

Setting Default Name Options for Tables and Columns

You can set default name options for all table metadata that is entered with a source designer wizard or a target designer wizard in SAS Data Integration Studio. These defaults apply to tables in SAS format or in DBMS format.

Defaults for table and column names can make it easier for users to enter the correct metadata for tables. Administrators still have to set name options on database libraries, and users should at least verify that the appropriate name options are selected for a given table.

Perform the following steps to set default name options for all table metadata that is entered with a source designer wizard or a target designer wizard in SAS Data Integration Studio:

1 Start SAS Data Integration Studio.

2 Open the metadata profile that specifies the repository where metadata for the tables is stored.

3 On the SAS Data Integration Studio desktop, select **Tools ▶ Options** from the menu bar. The Options window is displayed.

4 In the Options window, select the **`General`** tab.

5 On the **General** tab, select **Enable case-sensitive DBMS object names** to have source designers and the Target Table wizard support case-sensitive table and column names by default.

6 On the **General** tab, select **Enable special characters within DBMS object names** to have source designers and the Target Table wizard support special characters in table and column names by default.

7 Click **OK** to save any changes.

Maintaining Column Metadata

Problem

You want to work with the metadata for the columns of the tables or external files that you have registered in SAS Data Integration Studio. You might also need to perform common tasks such as sorting, reordering, and restoring the columns of the tables or external files. You might even need to attach a note or document to the metadata for a column.

Solution

You can use the Columns window, tab, or pane to maintain the metadata for columns in a table or external file. You can perform the following tasks on the metadata:

- Add Metadata for a Column
- Modify Metadata for a Column
- Import Metadata for a Column
- Delete Metadata for a Column
- Propagate New Columns
- Reorder Columns
- Sort Columns
- Restore the Order of Columns
- Save Reordered Columns
- Add and Maintain Notes and Documents for a Column

Note: Updates to any shared resource should be carefully planned to avoid unintended impacts on other applications that might use the same resource. For example, the metadata for a table might be used in a SAS Data Integration Studio job.

If a table is a target in a SAS Data Integration Studio job, changes to the metadata for that table are not reflected in the corresponding physical table until the next time that the job is executed. △

Tasks

Add Metadata for a Column

Perform the following steps to add a new column to the metadata for the current table:

1. Select a column from the following list that represents the new one that you want.
2. Click the **`New`** button. A row of default metadata describing the new column displays. The name of the column, **`Untitledn`**, is selected and ready for editing. The other attributes of the column have their default values:
   - Description: Blank
   - Length: 8
   - Type: Character
   - Summary Role: (None)
   - Sort Order: (None)
   - Informat: (None)
   - Format: (None)
   - Is Nullable: No
3. Change the name of the column to give it a meaningful name.
4. Change the values of other attributes for the column, as described in "Modify Metadata for a Column" on page 100.
5. Click **`Apply`** or **`OK`**.

For information about the implications of adding metadata to a column, see the note at the end of "Import Metadata for a Column" in Table 5.6 on page 101.

Modify Metadata for a Column

To modify the metadata for a column in the current table, select the attribute, make the change, and then click **Apply** or **OK**. The following table explains how to change each type of attribute.

Table 5.5 Column Metadata Modifications

Attribute	Instructions
Name	Perform the following steps to enter a name: **1** Double-click the current name to make it editable. **2** Enter a new name of 32 characters or fewer. **3** Press the **Enter** key.
Description	Perform the following steps to enter a description **1** Double-click in the **Description** field. **2** Edit the description, using 200 characters or fewer. **3** Press the **Enter** key.
Length	Perform the following steps to enter the column length: **1** Double-click the current length. **2** Enter a new length. A numeric column can be from 3 to 8 bytes long (2 to 8 in the z/OS operating environment). A character column can be 32,767 characters long. **3** Press the **Enter** key.
Type	Perform the following steps to enter the data type: **1** Double-click the current value to display the drop-down list arrow. **2** Click the arrow to make a list of valid choices appear. **3** Select a value from the list.
Summary Role	Perform same steps as for type.
Sort Order	Perform same steps as for type.
Informat	Perform the following steps to enter an informat: **1** Double-click the current value to display the drop-down list arrow. **2** Click the arrow to make a list of valid choices appear and then select a value from the list, or type in a new value and press the **Enter** key.
Format	Perform same steps as for informat.
Is Nullable	Perform same steps as for type.

You can also make a value editable by tabbing to it and pressing **F2** or any alphanumeric key. For information about the implications of modifying metadata for a column, see the note at the end of “Delete Metadata for a Column” in Table 5.6 on page 101.

Perform Additional Operations on Column Metadata

The following table describes some additional operations you can perform on column metadata.

Table 5.6 Additional Operations on Column Metadata

If you want to	Do this
Delete Metadata for a Column	Perform the following steps to delete the metadata for a column in the current table: **1** Select a column. **2** Click **`Delete`**. *Note:* When you modify or delete the metadata for a column in a table and that table is used in a SAS Data Integration Studio job, you might also have to make the same modifications to other tables in the job. For example, if you change the data type of a column and that table is used as a source in a job, then you need to change the data type of that column in the target table and in the temporary work tables in the transformations in that job. Changes to column metadata in SAS Data Integration Studio do not appear in the physical table unless you select **`Drop Table`** in the **`Load Technique`** tab of the Loader transformation that loads the current table. △
Import Metadata for a Column	Perform the following steps to import column metadata that has been added to the repositories that are specified in your current metadata profile: **1** Click **`Import Column`** to access the Import Column. **2** Select one or more columns from the **`Available Columns`** field in the Import Column window. **3** Select the right arrow to move the selected columns into the **`Selected Columns`** field. **4** Reorder the columns in the **`Selected Columns`** field by selecting columns and clicking the **`Moves select items up`** or **`Moves select items down`** arrows. **5** Click **`OK`** to import the columns into the table. Be aware of the following implications if you add or import metadata for a column: □ You might need to propagate that column metadata through the job or jobs that include the current table, as described in “Propagate Metadata for New Columns” in this table. □ The new column does not appear in the physical table unless you select **`Drop Table`** in the **`Load Technique`** tab of the Loader transformation that loads the current table.

If you want to	Do this
Propagate Metadata for New Columns	Perform the following steps to propagate a new column through the temporary work tables in the transformations in a job. In this way, the new column is propagated to the next target in the process flow diagram: **1** As necessary, open the job in the SAS Data Integration Studio Process Editor by double-clicking the job in the Project tree or another tree on the desktop. **2** In the Process Editor, add or import the column in all source and target tables that need that column, as described the earlier portions of this section. **3** Select **Process ▶ Propagate** from the menu bar and click **`Yes`** in the pop-up window. Selecting **`Propagate`** adds the new column to all of the temporary work tables that use that column. It also adds mappings that connect these work tables to the next targets. *Note:* □ Any mappings that were removed before selecting **`Propagate`** are restored after you select **`Propagate`**. □ If you add or import columns into a source and if that source is used in other jobs, you should consider propagating that change through those other jobs. △
Reorder Columns	You can rearrange the columns in a table (without sorting them) by (1) dragging a column to a new location or (2) using the arrow buttons at the bottom of the window. Drag a column to a new location by dragging the column-number cell. Perform the following steps to move a column or columns using the arrow buttons: **1** Select one or more columns. **2** Click the **`Moves columns up`** arrow to move the columns up, or click the **`Moves columns down`** arrow to move them down.
Restore the Order of Columns	Click the column number heading to restore all of the columns to their original order.

If you want to	Do this
Save Reordered Columns	Some windows allow you to change the default order of columns. Then, you can save that new order in the metadata for the current table or file. If you can save reordered columns before you exit the current window, SAS Data Integration Studio displays a dialog box that asks if you want to save the new order.
Sort Columns	You can sort the columns in a table based on the value of any column attribute (such as **`Name`** or **`Description`**) in either ascending or descending order. For example, you can sort the columns in ascending order by name by left-clicking the **`Name`** heading. To sort the columns in descending order by name, you can click the same heading a second time.

Add and Maintain Notes and Documents for a Column

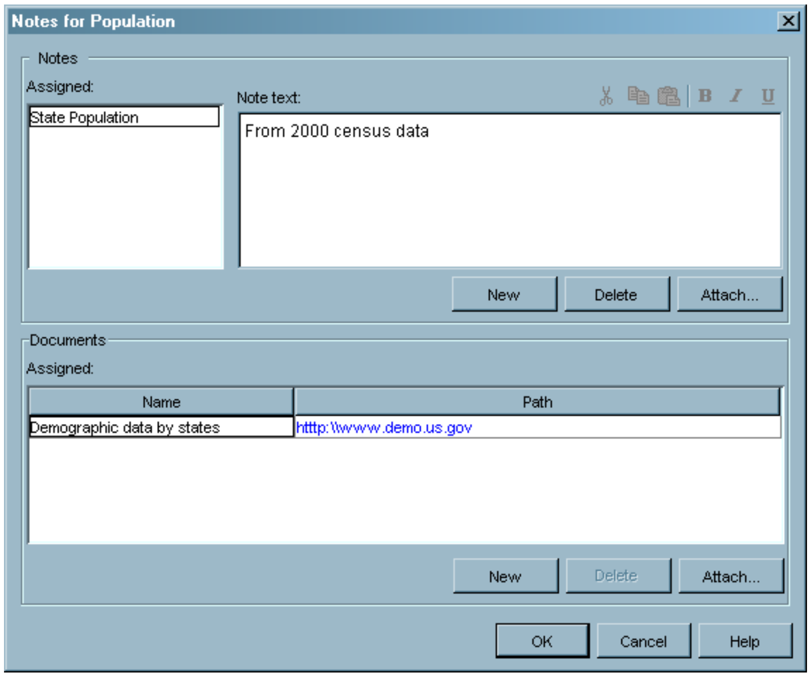
The Columns window, tab, or pane enables you to attach text notes, and documents produced by word processors, to the metadata for a table column. Such a note or document usually contains information about the table column or the values stored in that column.

Note: If a column currently has notes or documents associated with it, you can see a notes icon to the left of the column name. △

To add a note or document to a column, modify an existing note or document, or remove an existing note or document, you can use the Notes window. Follow these steps to get to this window:

1. Right-click the column you want to work with. Then, click **`Notes`** on the pop-up menu to access the Notes window for the selected column.
2. Perform one or more of the following tasks in the **`Notes`** group box:
   - □ Click **`New`** to create a new note. Enter a title in the **`Assigned`** field and the text of the note in the **`Note text`** field. Use the editing and formatting tools at the top of the window if you need them.
   - □ Click the name of an existing note in the **`Assigned`** field to review or update the content in the **`Note text`** field.
   - □ Click **`Delete`** to delete the note.
   - □ Click **`Attach`** to access the Select Additional Notes window and attach an additional note to the column.
3. Perform one or more of the following steps in the **`Documents`** group box:
   - □ Click **`New`** to attach a new document to the note. Enter a title in the **`Name`** field. Then, enter a path to the document in the **`Path`** field.
   - □ Click the name of an existing document in the **`Name`** field to review or update the path in the **`Path`** field.
   - □ Click **`Delete`** to delete the document.
   - □ Click **`Attach`** to access the Select Additional Documents window and attach an additional document to the column. The following display depicts a sample of a completed Notes window.

Display 5.3 Completed Notes Window



State Population

From 2000 census data

Name	Path
Demographic data by states	htttp:\\www.demo.us.gov

Identifying and Maintaining Key Columns

Problem

You want to identify existing columns in a table as primary, unique, or foreign keys, which are defined as follows:

- □ Primary key: one or more columns that are used to uniquely identify a row in a table. A table can have only one primary key. One or more columns in a primary key cannot contain null values.
- □ Foreign key: one or more columns that are associated with a primary key or unique key in another table. A table can have one or more foreign keys. A foreign key is dependent upon its associated primary or unique key. In other words, a foreign key cannot exist without that primary or unique key.
- □ Unique key: one or more columns that can be used to uniquely identify a row in a table. A table can have one or more unique keys. Unlike a primary key, a unique key can contain null values.

You also need to modify and delete existing key columns. If you do need to generate new key values, see Chapter 19, "Working with Slowly Changing Dimensions," on page 343.

Solution

You can use the **Keys** tab in the properties window for a table to identify key columns in the table, modify existing keys, and delete keys. You can perform the following tasks when you work with keys:

- □ Identify a Primary or Unique Key Column

- Add a Foreign Key Column
- Add a Column to a Primary or Unique Key
- Remove a Column from a Primary or Unique Key
- Rearrange the Columns in a Multi-Column Key
- Rename a Key
- Delete a Key

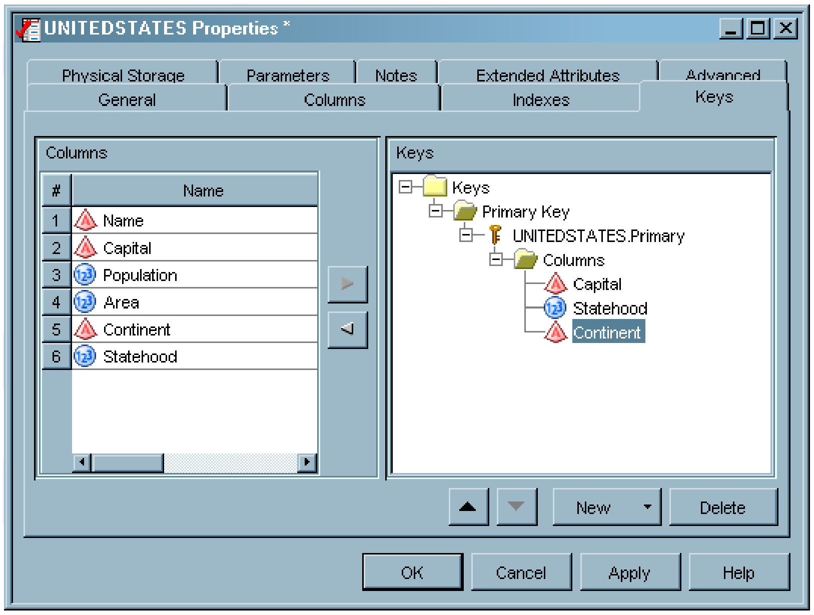
Tasks

Identify a Column as a Primary or Unique Key

Perform the following steps to designate existing columns as primary keys or unique keys in a table:

1. Access the **Keys** tab on the properties sheet for a table.
2. Select the one or more columns from the Columns table that you want to serve as the primary or unique key.
3. Click on the down-arrow portion of the **New** combination button, and select **Primary Key** or **Unique Key**, depending on the type of key that you are creating. The new nodes display in the tree in **Keys** field. The nodes include the key and the columns in the key.
4. Click **Apply** or **OK**. A sample primary node is depicted in the following display.

Display 5.4 Completed Primary Node



#	Name
1	Name
2	Capital
3	Population
4	Area
5	Continent
6	Statehood

Note: By default, the keys are named as follows:

- *TableName*.Primary indicates a primary key.
- Unique keys are named *TableName*.Unique Key#, such as EMPLOYEE_JAN_DTM.Unique Key1 for the first unique key.

△

Add a Foreign Key Column

Perform the following steps to identify a foreign key in a table:

1 Click on the down-arrow portion of the **New** combination button, and select **Foreign Key**. This displays the Foreign Key Wizard window.

2 In the wizard, click **Help** for guidance in creating the foreign key. When you have finished running the wizard, additional nodes display in the tree in the **Keys** field.

Note: For a complete example of how to run the Foreign Key Wizard, see the "Example: Running the Foreign Key Wizard" topic in SAS Data Integration Studio help. △

Add a Column to a Primary or Unique Key

Perform the following steps to add a new column to an existing primary or unique key:

1 From the columns table, select the column that you want to add to the primary or unique key.

2 In the keys tree, select either the folder that contains the key columns or one of the columns in the key.

3 Click **Add** or drag and drop one or more columns into the **Columns** field in the keys tree. If you selected the folder, the key that you add becomes the first column in the list of columns that make up the key. If you selected a column, the new column is added below the selected column.

4 Click **OK** or **Apply**.

Remove a Column from a Primary or Unique Key

Perform the following steps to remove a column from an existing primary or unique key:

1 Select a column that you want to remove in the tree in the **Keys** field.

2 Click **Remove**, or press the **Delete** key on your keyboard.

3 Click **OK** or **Apply**.

Rearrange the Columns in a Multi-Column Key

Perform the following steps to rearrange the columns in a multi-column key:

1 Select a column that you want to move up or down in a list of columns in the tree in the **Keys** field.

2 Click **Move a column up** to move the column up in the list or click **Move a column down** to move the column down.

3 Click **OK** or **Apply**.

Rename a Key

Perform the following steps to rename a key:

1 Right-click on the **Key** folder and click **Rename** in the pop-up menu. This makes the key name editable.

2 Enter the new name, and press the **Enter** key.

3 Click **OK** or **Apply**.

Delete a Key

Perform the following steps to delete a key:

1. Click the folder that represents the key that you want to delete. This action selects the key.
2. Click the **`Delete`** button, or press the **`Delete`** key on your keyboard.
3. Click **`OK`** or **`Apply`**.

Maintaining Indexes

Problem

You want to create a new index or modify or delete an existing index. See the "Indexes Window/Tab" topic in SAS Data Integration Studio help for more information about indexes.

Solution

You can use the **`Indexes`** tab on the properties window for a table.

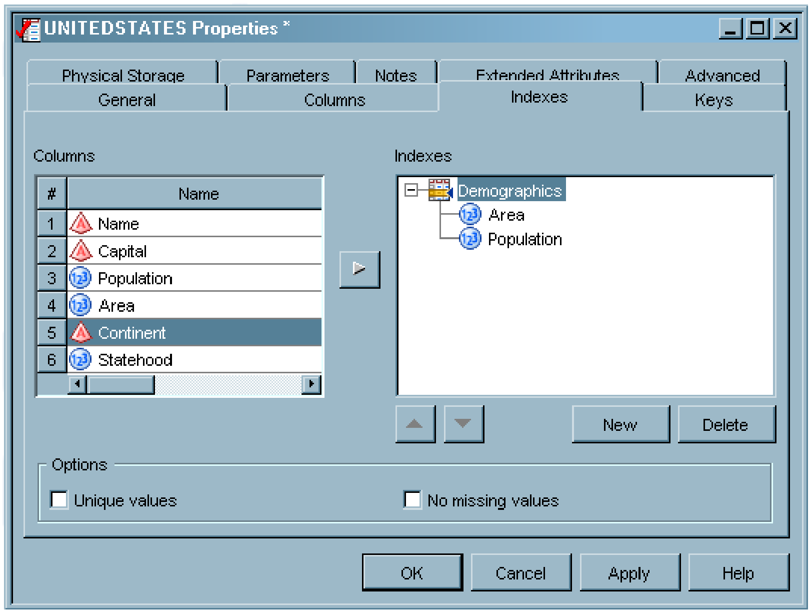
Tasks

Create a New Index

Perform the following steps to create a new index in the **`Indexes`** tab:

1. Click **`New`**. A folder displays in the tree in the **`Indexes`** field. This folder represents an index and has an appropriate default name. The name is selected for editing. You can rename the index to a more appropriate value by typing over the existing name and pressing the **`Enter`** key.
2. Perform the following steps to add one or more columns to the index:
   - Drag a column name from the **`Columns`** field to an index folder in the **`Indexes`** field.
   - Select a column name in the **`Columns`** field. Then, select an **`Index`**. Finally, click **`Add`**.
3. Click **`OK`**. The following display depicts a sample index.

Display 5.5 Sample Completed Index



#	Name
1	Name
2	Capital
3	Population
4	Area
5	Continent
6	Statehood

Indexes:
- Demographics
  - Area
  - Population

New Delete

Options
- [ ] Unique values
- [ ] No missing values

Note: If you add one column to the index, you create a simple index. If you add two or more columns, you create a composite index. If you want the index to be unique, select the index name in the **`Indexes`** field, and then select the **`Unique values`** check box. Finally, if you are working with a SAS table and want to ensure that the index contains no missing values, check the **`No missing values`** check box. △

Delete an Index or a Column

Perform the following steps to delete an index or to delete a column from an index in the Indexes window or tab:

1 Select the index or column in the tree in the **`Indexes`** field.

2 Click the **`Delete`** button, or press the **`Delete`** key on your keyboard.

3 Click **`OK`** or **`Apply`**.

Rearrange the Columns in an Index

You can reorder the columns for composite indexes, which contain more than one column. Perform the following steps to move a column up or down in the list of index columns in the Indexes window or tab:

1 Select the column that you want to move in the tree in the **`Indexes`** field.

2 Use the **`Move columns up in an index`** and **`Move columns down in an index`** buttons to move the column up or down.

3 After you have arranged the columns as you want them, click **`OK`** or **`Apply`**.

Note: It is generally best to list the column that you plan to search the most often first. △

Browsing Table Data

Problem

You want to display data in a SAS table or view, in an external file, in a temporary output table displayed in a process flow diagram, or in a DBMS table or view that is part of a SAS library for DBMS data stores.

Solution

You can use the browse mode of the View Data window, provided that the table, view, or external file is registered in a current metadata repository and exists in physical storage. You can browse temporary output tables until the Process Designer window is closed or the current server session is ended in some other way.

Transformations in a SAS Data Integration Studio job can create temporary output tables. If these temporary tables have not been deleted, you can also use the browse mode to display the data that they contain. The transformation must have been executed at least once for the temporary output tables to exist in physical storage.

Tasks

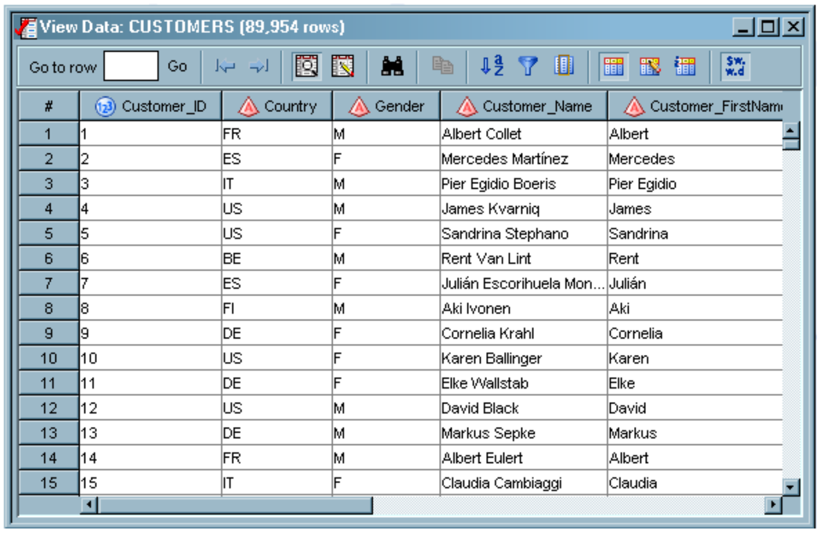
Use Browse Mode in the View Data Window

Perform the following steps to browse data in the View Data window:

1 In the Inventory tree, other tree view, or Process Designer window, right-click the metadata object for the table, view, external file, temporary output, or transformation. Then, select **`View Data`** from the pop-up menu.

2 Enter the appropriate user ID and password, if you are prompted for them. The information in the table, view, or external file displays in the View Data window, as shown in the following display.

Display 5.6 View Data Window in Browse Mode



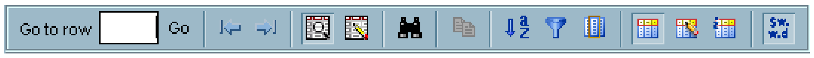
#	Customer_ID	Country	Gender	Customer_Name	Customer_FirstNam
1	1	FR	M	Albert Collet	Albert
2	2	ES	F	Mercedes Martínez	Mercedes
3	3	IT	M	Pier Egidio Boeris	Pier Egidio
4	4	US	M	James Kvarniq	James
5	5	US	F	Sandrina Stephano	Sandrina
6	6	BE	M	Rent Van Lint	Rent
7	7	ES	F	Julián Escorihuela Mon...	Julián
8	8	FI	M	Aki Ivonen	Aki
9	9	DE	F	Cornelia Krahl	Cornelia
10	10	US	F	Karen Ballinger	Karen
11	11	DE	F	Elke Wallstab	Elke
12	12	US	M	David Black	David
13	13	DE	M	Markus Sepke	Markus
14	14	FR	M	Albert Eulert	Albert
15	15	IT	F	Claudia Cambiaggi	Claudia

The title bar of the View Data window displays the name of the object that is being viewed and the total number of rows.

Browse Functions

The browse mode of the View Data window contains a group of functions that enable you to customize how the data in the window is displayed. These functions are controlled by the view data toolbar, as shown in the following display.

Display 5.7 View Data Browse Toolbar



Perform the tasks listed in the following table to customize the data display:

Table 5.7 Browse Functions in the View Data Window

If you want to	Do this
Navigate within the data	Perform the following steps: □ Enter a row number in the **Go to row** field and click **Go to row** to specify the number of the first row that is displayed in the table. □ Click **Go to first row** to navigate to the first row of data in the View Data window. □ Click **Go to last row** to navigate to the last row of data in the View Data window.
Select a View Data window mode	Perform the following steps: □ Click **Switch to browse mode** to switch to the browse mode. □ Click **Switch to edit mode** to switch to the edit mode. *Note:* The **Switch to browse mode** and **Switch to edit mode** buttons are displayed only for SAS tables. △
Copy one or more rows of data into the copy buffer	Perform the following steps: □ Highlight one or more rows of data. Then, click **Copy** to copy the selected text into the copy buffer.

If you want to	Do this
Manipulate the data displayed in View Data window	Perform the following steps: □ Click **`Launch search screen`**. Then, use the search toolbar to search for string occurrences in the data set that is currently displayed in the View Data window. □ Click **`Launch sort screen`**. Then, use the **`Sort By Columns`** tab in the View Data Query Options window to specify a sort condition on multiple columns. The sort is performed on the data set that is currently displayed in the View Data window. □ Click **`Launch filter screen`**. Then, use the **`Filter`** tab in the View Data Query Options window to specify a filter clause on the data set that is currently displayed in the View Data window. This filter clause is specified as an SQL WHERE clause that is used when the data is fetched. □ Click **`Launch column subset screen`**. Use the **`Columns`** tab in the View Data Query Options window to select a list of columns that you want to see displayed in the View Data window. You can create a subset of the data currently displayed in the View Data window by selecting only some of the available columns in the **`Columns`** field. The redrawn View Data window will include only the columns that you select here on the **`Columns`** tab.
Determine what is displayed in the column headers	You can display any combination of column metadata, physical column names, and descriptions in the column headers. □ Click **`Show column name in column header`** to display physical column names in the column headings. □ Click **`Show description in column header`** to display optional descriptions in the column headers. □ Click **`Show metadata name in column header`** to display optional column metadata in the column headers. This metadata can be entered in some SAS Business Intelligence applications, such as the SAS Information Mapping Studio.
Determine whether metadata formats are applied	Perform the following steps: □ Click **`Apply metadata formats`** to toggle between showing formatted and unformatted data in the View Data window.

To sort columns and perform related tasks, right-click on a column name and select an appropriate option from the pop-up menu. To refresh the view, select **View ▶ Refresh** from the menu bar. For other details about using this window, see "Pop-Up Menu Options in the View Data Window" in the SAS Data Integration Studio help.

To set options for the View Data window, select **File ▶ Options** from the SAS Data Integration Studio menu bar to display the Options window. Then, click the **`View Data`** tab. See “Specifying Browse and Edit Options for Tables and External Files” on page 116 for information about the available options.

To access the View Data drop-down menu, click anywhere in the View Data window and click the View Data heading on the SAS Data Integration Studio menu bar. For information about the available menu items, see the View Data section in the "Menu Bar" topic in the SAS Data Integration Studio help.

Additional Information

General information about the View Data window is available in the "View Data Window" Help topic in SAS Data Integration Studio. For detailed information about specific usage issues, see the "Usage Notes for the View Data Window" Help topic.

Editing SAS Table Data

Problem

You want to edit SAS table data that is displayed in the View Data window.

Solution

You can use the edit mode of the View Data window to perform simple editing operations in a SAS table. The editing mode is enabled only on SAS tables that are stored in a BASE engine library and are assigned on the workspace server. The entity that you want to edit must be registered in a current metadata repository, and it must exist in physical storage. If you are working under change management, you must check out the entity before you can edit it in the View Data window.

Tasks

Use Edit Mode in the View Data Window

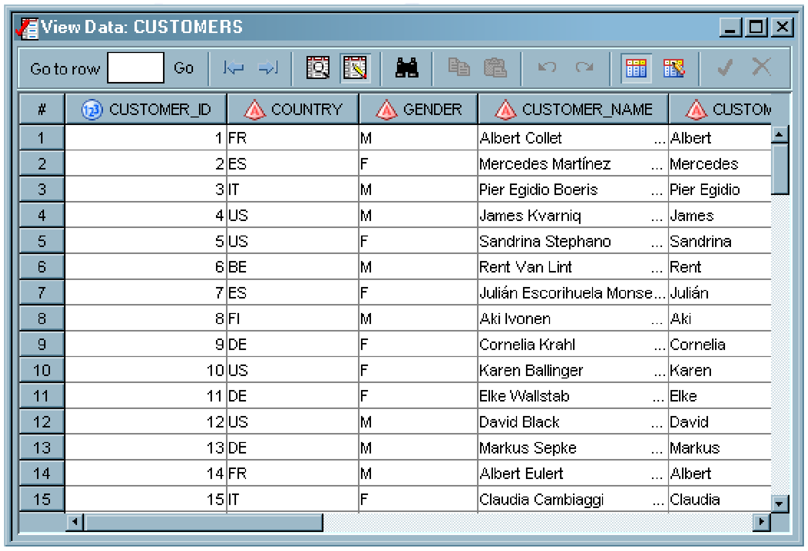
Perform the following steps to edit data for a SAS table in the View Data window:

1 In the Inventory tree, other tree view, or Process Designer window, right-click the metadata object for a SAS table. Then, select **`View Data`** from the pop-up menu.

2 Enter the appropriate user ID and password, if you are prompted for them. The information in the table displays in the browse mode of the View Data window.

3 Click **`Switch to edit mode`** on the view data toolbar. The View Data window displays in edit mode, as shown in the following display.

Display 5.8 View Data Window in Edit Mode



#	CUSTOMER_ID	COUNTRY	GENDER	CUSTOMER_NAME	CUSTOM
1	1	FR	M	Albert Collet ...	Albert
2	2	ES	F	Mercedes Martínez ...	Mercedes
3	3	IT	M	Pier Egidio Boeris ...	Pier Egidio
4	4	US	M	James Kvarniq ...	James
5	5	US	F	Sandrina Stephano ...	Sandrina
6	6	BE	M	Rent Van Lint ...	Rent
7	7	ES	F	Julián Escorihuela Monse...	Julián
8	8	FI	M	Aki Ivonen ...	Aki
9	9	DE	F	Cornelia Krahl ...	Cornelia
10	10	US	F	Karen Ballinger ...	Karen
11	11	DE	F	Elke Wallstab ...	Elke
12	12	US	M	David Black ...	David
13	13	DE	M	Markus Sepke ...	Markus
14	14	FR	M	Albert Eulert ...	Albert
15	15	IT	F	Claudia Cambiaggi ...	Claudia

The title bar of the View Data window displays the name of the object that is being viewed.

4 Double-click inside a cell and then change the data in the cell. Click **`Commit edit row`** to commit the change to the database. Of course, you must have operating system access for the file for the change to be saved.

5 Click **`Undo last action`** to reverse the change that you just made. (You can click **`Redo last action`** to return to the changed version of the cell.) Note that you can undo only the last operation because only a single level of undo is supported. So if multiple rows have been deleted or pasted, only the last row affected can be undone. Similarly, you can redo only your latest undo.

6 Click a row number to select the row. Click **`Copy`** to copy the row into the buffer.

7 Click **`Go to last row`** to move to the last row in the table.

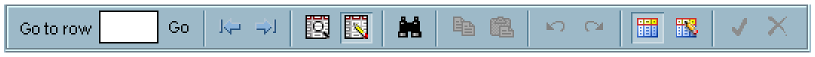
8 Click in the row marked by the New Row icon at the end of the View Data window. The New Row icon changes to the Editing Row icon. Click **`Paste`** to paste the copied data into the row.

9 Click **`Delete selected rows`** to delete the pasted data and remove the row from the table.

Edit Tasks

The edit mode of the View Data window contains a group of functions that enable you to customize how the data in the window is displayed. These functions are controlled by the view data toolbar, as shown in the following display.

Display 5.9 View Data Edit Toolbar



Perform the tasks listed in the following table to edit the data displayed:

Table 5.8 Edit Functions in the View Data Window

If you want to	**Do this**
Navigate within the data	Perform the following steps: □ Enter a row number in the **`Go to row`** field and click **`Go to row`** to specify the number of the first row that is displayed in the table. □ Click **`Go to first row`** to navigate to the first row of data in the View Data window.
Select a View Data window mode	Perform the following steps: □ Click **`Switch to browse mode`** to switch to the browse mode. □ Click **`Switch to edit mode`** to switch to the edit mode. *Note:* The **`Switch to browse mode`** and **`Switch to edit mode`** buttons are displayed only for SAS tables. △
Copy or paste data	Perform the following steps: □ Highlight one or more rows of data. Then, click **`Copy`** to copy the selected text into the copy buffer. □ Place the cursor in the row where you want to place the data. Then, click **`Paste`** to paste the data into the table.
Search the data displayed in View Data window	Perform the following steps: □ Click **`Launch search screen`**. Then, use the search toolbar to search for string occurrences in the data set that is currently displayed in the View Data window.
Undo or redo editing operations	Perform the following steps: □ Click **`Undo last action`** to reverse the most recent editing operation. □ Click **`Redo last action`** to restore the results of the most recent editing operation.

If you want to	Do this
Determine what is displayed in the column headers	You can display any combination of column metadata, physical column names, and descriptions in the column headers. □ Click `Show column name in column header` to display physical column names in the column headers. □ Click `Show description in column header` to display displays optional descriptions in the column headers.
Commit or delete editing changes	Perform the following steps: □ Click `Commit edit row` to commit the changes that you have made to the currently edited row. □ Click `Delete edit row` to delete the changes that you have made to the currently edited row.

To hide, show, hold, and release columns, right-click on a column name and select an appropriate option from the pop-up menu. To refresh the view, select **View ▶ Refresh** from the menu bar. For other details about using this window, see "Pop-Up Menu Options in the View Data Window" in the SAS Data Integration Studio help.

To set options for the View Data window, select **File ▶ Options** from the SAS Data Integration Studio menu bar to display the Options window. Then, click the `View Data` tab. See “Specifying Browse and Edit Options for Tables and External Files” on page 116 for information about the available options.

To access the View Data drop-down menu, click anywhere in the View Data window and click the View Data heading on the SAS Data Integration Studio menu bar. For information about the available menu items, see the View Data section in the "Menu Bar" topic in the SAS Data Integration Studio help.

Additional Information

General information about the View Data window is available in the "View Data Window" Help topic in SAS Data Integration Studio. For detailed information about specific usage issues, see the "Usage Notes for the View Data Window" Help topic.

Using the View Data Window to Create a SAS Table

Problem

You want to create a new SAS table. This method can be used to create small tables for testing purposes.

Solution

Use the create table function of the View Data window. This function enables you to create a new SAS table based on metadata that you register by using the Target Table wizard.

Tasks

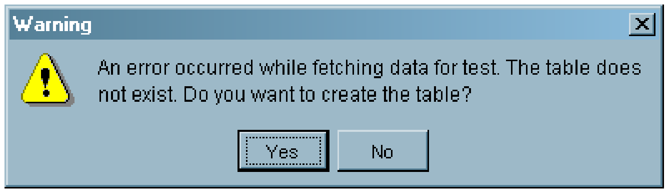
Using the Create Table Function in the View Data Window

Perform the following steps to create a new table in the View Data window:

1 Create the metadata for a new SAS table in the Target Table wizard. Select the columns that you need from existing tables.

2 Right-click the newly registered table and click **`View Data`**. The dialog box in the following display is shown.

Display 5.10 Create Table Dialog Box



An error occurred while fetching data for test. The table does not exist. Do you want to create the table?

3 Click **`Yes`** to create the table in the SAS library that you specified in the metadata for the table. The table is opened in edit mode.

Additional Information

General information about the View Data window is available in the "View Data Window" Help topic in SAS Data Integration Studio. For detailed information about specific usage issues, see the "Usage Notes for the View Data Window" Help topic.

Specifying Browse and Edit Options for Tables and External Files

Problem

You want to set options that control how tables and external files are processed in the browse and edit modes in the View Data window.

Solution

You can use the **`View Data`** tab in the Options window to specify options for the View Data window. The options that you set on the **`View Data`** tab are applied globally. The tab is divided into the **`General`** group box, the **`Column Headers`** group box, the **`Format`** group box, the **`Search`** group box, and the **`Editing`** group box.

Tasks

Set General Options

The **`General`** group box contains the following items:

Table 5.9 General Options

Option	Description
Clear Query Options when refreshing	Clears any options that you set on the query when you refresh the data.
Prompt for long-running navigation operation	Determines whether the user is prompted to decide whether the View Data query should proceed with a long-running navigation operation. If this option is selected, the prompt is displayed whenever the row count of the table is either not known or greater than 100,000. If the option is deselected, the navigation operation proceeds without the warning prompt.

Set Column Header Options

The **`Column Headers`** group box contains the following items:

Table 5.10 Column Headers Options

Option	Description
Show column name in column header	Displays physical column names in the column headers.
Show column description in column header	Displays optional descriptions in the column headers.
Show column metadata name in column header	Displays optional column metadata names in the column headers. This metadata can be entered in some SAS Business Intelligence applications, such as the SAS Information Mapping Studio.

Note: You can display any combination of column metadata, SAS column names, and descriptions in the column headers by selecting the combination of check boxes that are required to get the result that you want. △

Set Format Options

The **Format** group box contains the following items:

Table 5.11 Format Options

Option	Description
Apply formats	When selected, displays formatted data in the View Data window. This option applies the permanent formats that are specified for the data when the data set is created. Deselect the check box to view unformatted data in the View Data window.
Apply metadata formats	When selected, uses metadata formats for formatted data that is displayed in the View Data window. These formats are specified in the metadata for the data set.

Set Search Options

The **Search** group box contains the following items:

Table 5.12 Search Options

Option	Description
Recently specified search string (entries)	Specifies the number of recently searched strings that are displayed when you click the drop-down menu in the **Search for** field.
Ignore invalid column names	When selected, ignores any invalid column names that are entered into the search.

Set Editing Options

The **Editing** group box contains the following items:

Table 5.13 Editing Options

Option	Description
Allow editing of SCD2 tables without prompting	Determines whether a warning dialog box that states that edits to Slowly Changing Dimension (SCD) tables will cause the SCD to no longer be valid is displayed.
Always delete rows without prompting	Determines whether a warning dialog box is displayed before rows are deleted.
On multi-row operation errors	When one or more errors occur in a multi-row editing operation, determines whether the user is prompted, errors are ignored and the operation is continued, or the operation is canceled.
Default date format	Specifies a default format for date values.
Default datetime format	Specifies a default format for datetime values.

CHAPTER
6

Working with External Files

About External Files

An external file is a file that is maintained by the machine operating environment or by a software product other than SAS. A flat file with comma-separated values is one example.

SAS Data Integration Studio provides the following source designer wizards that enable you to create metadata objects for external files:

- Delimited External File wizard: Use for external files in which data values are separated with a delimiter character. You can specify multiple delimiters, nonstandard delimiters, missing values, and a multi-line record.
- Fixed Width External File wizard: Use for external files in which data values appear in columns that are a specified number of characters wide. You can specify non-contiguous data.
- User Written External File wizard: Use for complex external files that require user-written SAS code for data access

You can use the external file source designer wizards to do the following:

- display a raw view of the data in the external file
- display a formatted view of the data in the external file, as specified in the SAS metadata for that file
- display the SAS DATA step and SAS INFILE statement that the wizard generates for the selected file
- display the SAS log for the code that is generated by the wizard

- specify options for the SAS INFILE statement that is generated by the wizard, such as National Language Support (NLS) encoding
- override the generated SAS INFILE statement with a user-written statement
- supply a user-written SAS DATA step to access an external file

The most common tasks that you can perform with external files are listed in the following table.

Table 6.1 Common External File Tasks

If you want to	**Do this**
Access external files on a server	The external file source designers prompt you to specify the physical path to an external file and to select a server that can resolve that path. You can select any SAS Application Server, FTP server, or HTTP/HTTPS server that is defined in a current metadata repository. For details, see "Accessing an External File With an FTP Server or an HTTP Server" on page 136.
Generate column metadata	The external file wizards enable you to automatically generate column metadata based on the structure of the external file, a guessing algorithm, and some parameters that you supply. The guessing algorithm supports numeric, character, date/time, and monetary formats. Defaults can be supplied for numeric and character formats. The external file wizards enable you to import column metadata in the following ways: □ from the metadata for a table or an external file in a current metadata repository. □ from an external format file. For information, see the "Using External Format Files" topic in SAS Data Integration Studio help. □ from a COBOL copybook. For information, see "Registering a COBOL Data File That Uses a COBOL Copybook" on page 139. □ from a column heading in the external file. You can also manually enter the metadata for each column.

If you want to	Do this
View and update external files	You can update the metadata for an external file, as described in the "Viewing or Updating Metadata for Other Objects" topic in SAS Data Integration Studio help. You can display a raw view of the data in an external file or display a formatted view of this data, as specified in the metadata for the file. For details, see the "Viewing Data for Tables and Files" topic in SAS Data Integration Studio help.
Use external files in jobs	The metadata for an external file can be used in a SAS Data Integration Studio job that reads data from the file or writes data to the file. For more information about jobs, see “About Tables” on page 85. To support jobs that include external files, two new transformations have been added to the Process Library: □ File Reader: Reads an external file and writes to a target. Added automatically to a process flow when an external file is specified as a source. Executes on the host where the external file resides, as specified in the metadata for the external file. □ File Writer: Reads a source and writes to an external file. Added automatically to a process flow when an external file is specified as a target. Executes on the host where the external file resides, as specified in the metadata for the external file. The **`Mapping`** tab in the properties window for these two transformations enables you to define derived mappings.

Registering a Delimited External File

Problem

You want to create metadata for a delimited external file so that it can be used in SAS Data Integration Studio.

Solution

Use the delimited external file source designer to register the file. The source designer enables you to create metadata for external files that contain delimited data. This metadata is saved to a SAS Metadata Repository.

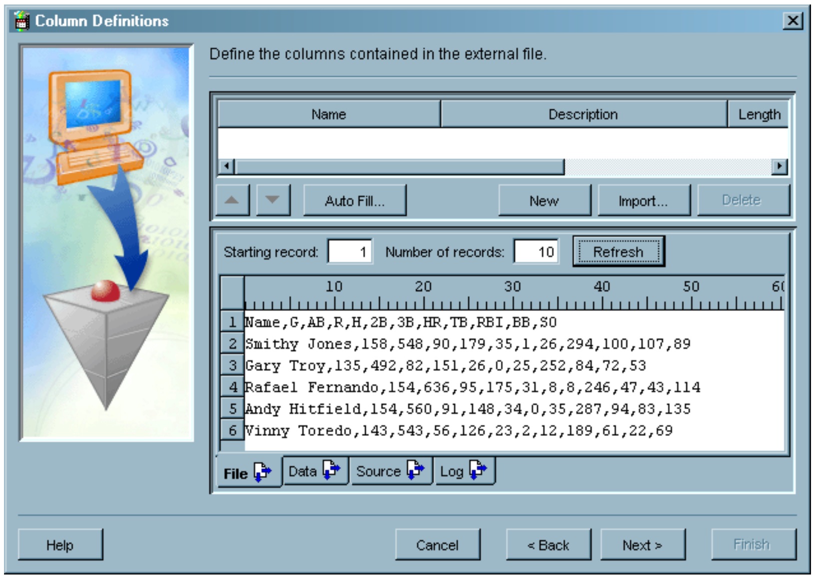
Tasks

Run the Delimited External File Source Designer

Perform the following steps to use one method to register an external file in the delimited external file source designer:

1. Open the Source Designer window. Open the **`External Files`** folder and click **`Delimited External File`**. Click **`Next`** to access the External File Location window.
2. If you are prompted, enter the user ID and password for the default SAS Application Server that is used to access the external file.
3. Specify the physical path to the external file in the **`File name`** field. Click **`Next`** to access the Delimiters and Parameters window.
4. Select the check box for the appropriate delimiter in the **`Delimiters`** group box. Accept the default values for the remaining fields, and click **`Next`** to access the Column Definitions window.
5. Click **`Refresh`** to view the raw data from the external file in the **`File`** tab in the view pane at the bottom of the window. Sample data is shown in the following display.

Display 6.1 Delimited Data in the File Tab



```
Name,G,AB,R,H,2B,3B,HR,TB,RBI,BB,SO
Smithy Jones,158,548,90,179,35,1,26,294,100,107,89
Gary Troy,135,492,82,151,26,0,25,252,84,72,53
Rafael Fernando,154,636,95,175,31,8,8,246,47,43,114
Andy Hitfield,154,560,91,148,34,0,35,287,94,83,135
Vinny Toredo,143,543,56,126,23,2,12,189,61,22,69
```

Note: If your external file contains fewer than 10 rows, a warning box will be displayed. Click **`OK`** to dismiss the warning window. △

6. Click **`Auto Fill`** to access the Auto Fill Columns window and populate preliminary data into the columns component of the Columns Definition window.
7. The first row in most external files is unique because it holds the column names for the file. Therefore, you should change the value that is entered in the **`Start record`** field in the **`Guessing records`** group box to **`2`**. This setting ensures that the guessing algorithm begins with the second data record in the external file. Excluding the first data from the guessing process yields more accurate preliminary data.

8 Accept all of the remaining default settings. Click **`OK`** to return to the Column Definitions window.

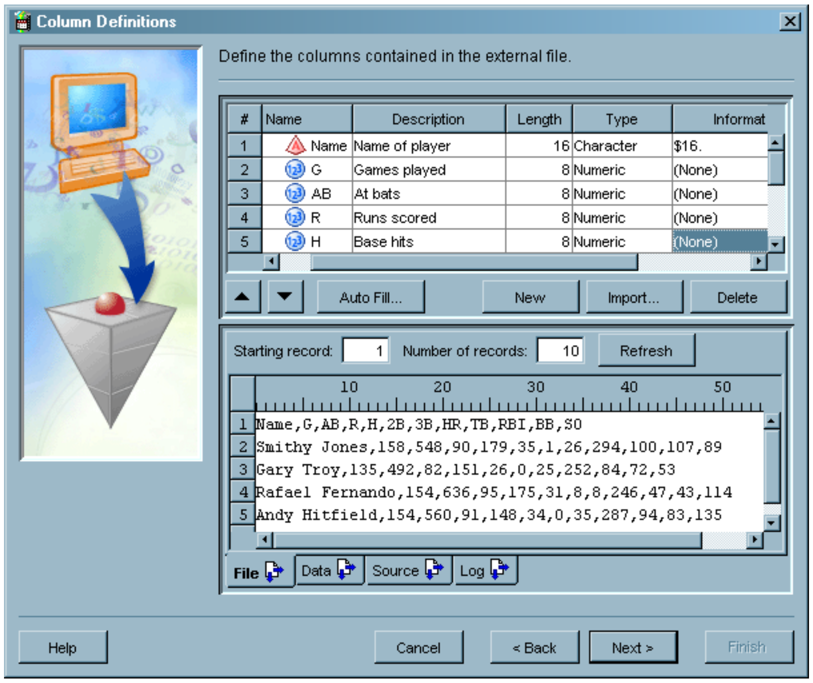
9 Click **`Import`** to access the Import Column Definitions window and the import function to simplify the task of entering column names.

10 Select the **`Get the column names from column headings`** in the field radio button, and keep the default settings for the fields underneath it. Click **`OK`** to save the settings and return to the Column Definitions window. The names from the first record in the external file are populated in the **`Name`** column. You now can edit them as needed.

Note: If you use the get column names from column headings function, the value in the **`Starting record`** field in the **`Data`** tab of the view pane in the Column Definitions window is automatically changed. The new value is one greater than the value in the **`The column headings are in file record`** field in the Import Column Definitions window. △

11 The preliminary metadata that is populated into the columns component usually includes column names and descriptions that are too generic to be useful for SAS Data Integration Studio jobs. Fortunately, you can modify the columns component by clicking in the cells that you need to change and entering the correct data. Enter appropriate values for the external file that you are registering. The following display depicts a sample completed Column Definitions window.

Display 6.2 Sample Completed Column Definitions Window



Define the columns contained in the external file.

#	Name	Description	Length	Type	Informat
1	Name	Name of player	16	Character	$16.
2	G	Games played	8	Numeric	(None)
3	AB	At bats	8	Numeric	(None)
4	R	Runs scored	8	Numeric	(None)
5	H	Base hits	8	Numeric	(None)

Starting record: 1 Number of records: 10

```
1 Name,G,AB,R,H,2B,3B,HR,TB,RBI,BB,SO
2 Smithy Jones,158,548,90,179,35,1,26,294,100,107,89
3 Gary Troy,135,492,82,151,26,0,25,252,84,72,53
4 Rafael Fernando,154,636,95,175,31,8,8,246,47,43,114
5 Andy Hitfield,154,560,91,148,34,0,35,287,94,83,135
```

12 To verify that the metadata you have entered is appropriate for the data in the external file, click the **`Data`** tab and then click **`Refresh`**. If the metadata matches the data, the data will be properly displayed in the **`Data`** tab. In the current example, the **`Data`** tab would look similar to the View Data window for the registered external file. If the data does not display properly, update the column metadata and click **`Refresh`** to verify that the appropriate updates have been made. To view the code that will be generated for the external file, click the **`Source`** tab. To view the SAS log for the generated code, click the **`Log`** tab. The code that is displayed in the **Source** tab is the code that will be generated for the current external file when it is included in a SAS Data Integration Studio job.

13 Click **Next** to access the Select Folder window. Select an appropriate Custom tree folder for the external file metadata.

14 Click **Next** to access the General window. Enter an appropriate name for the external file metadata. (If you leave the **Name** field blank, the file is identified by its path.) You can also enter an optional description.

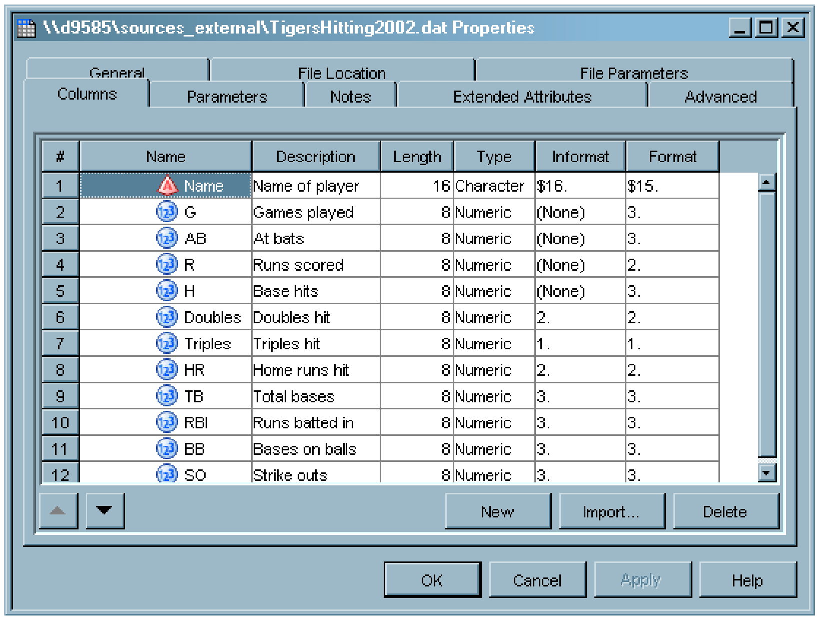
15 Click **Finish** to save the metadata and exit the delimited external file source designer.

View the External File Metadata

After you have generated the metadata for an external file, you can use SAS Data Integration Studio to view, and possibly make changes to, that metadata. For example, you might want to remove a column from a table or change the data type of a column. Any changes that you make to this metadata will not affect the physical data in the external file. However, the changes will affect the data that is included when the external table is used in SAS Data Integration Studio. Perform the following steps to view or update external file metadata:

1 Right-click the external file, and click **Properties**. Then, click the **Columns** tab. The Columns window is displayed, as shown in the following example.

Display 6.3 External File Columns Window



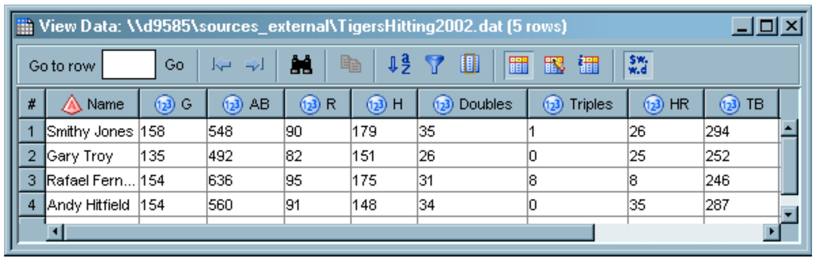
#	Name	Description	Length	Type	Informat	Format
1	Name	Name of player	16	Character	$16.	$15.
2	G	Games played	8	Numeric	(None)	3.
3	AB	At bats	8	Numeric	(None)	3.
4	R	Runs scored	8	Numeric	(None)	2.
5	H	Base hits	8	Numeric	(None)	3.
6	Doubles	Doubles hit	8	Numeric	2.	2.
7	Triples	Triples hit	8	Numeric	1.	1.
8	HR	Home runs hit	8	Numeric	2.	2.
9	TB	Total bases	8	Numeric	3.	3.
10	RBI	Runs batted in	8	Numeric	3.	3.
11	BB	Bases on balls	8	Numeric	3.	3.
12	SO	Strike outs	8	Numeric	3.	3.

2 Click **OK** to save any changes and close the properties window.

View the Data

Right-click the external file, and click **View Data**. The View Data window is displayed, as shown in the following example.

Display 6.4 External File Data in the View Data Window


View Data: \\d9585\sources_external\TigersHitting2002.dat (5 rows)

#	Name	G	AB	R	H	Doubles	Triples	HR	TB
1	Smithy Jones	158	548	90	179	35	1	26	294
2	Gary Troy	135	492	82	151	26	0	25	252
3	Rafael Fern...	154	636	95	175	31	8	8	246
4	Andy Hitfield	154	560	91	148	34	0	35	287

If the data in the external file displays correctly, the metadata for the file is correct and the table is available for use in SAS Data Integration Studio. If you need to review the original data for the file, right-click on its metadata object. Then, click **View File**.

Additional Information

General information about registering external files window is available in the "Managing External Files in SAS Data Integration Studio" Help topic in SAS Data Integration Studio. For detailed information about specific usage issues, see the "Usage Notes for External Files" Help topic.

Registering a Fixed-Width External File

Problem

You want to create metadata for a fixed-width external file so that it can be used in SAS Data Integration Studio.

Solution

Use the fixed-width external file source designer to register the file. The source designer enables you to create metadata for external files that contain fixed-width data. The metadata is saved to a SAS Metadata Repository.

You need to know the width of each column in the external file. This information might be provided in a document that describes the structure of the external file.

Tasks

Run the Fixed-Width External File Source Designer

Perform the following steps to use one method to register an external file in the fixed-width external file source designer:

1 Open the Source Designer window. Open the **External Files** folder and click **Fixed Width External File**. Click **Next** to access the External File Location window.

2 If you are prompted, enter the user ID and password for the default SAS Application Server that is used to access the external file.

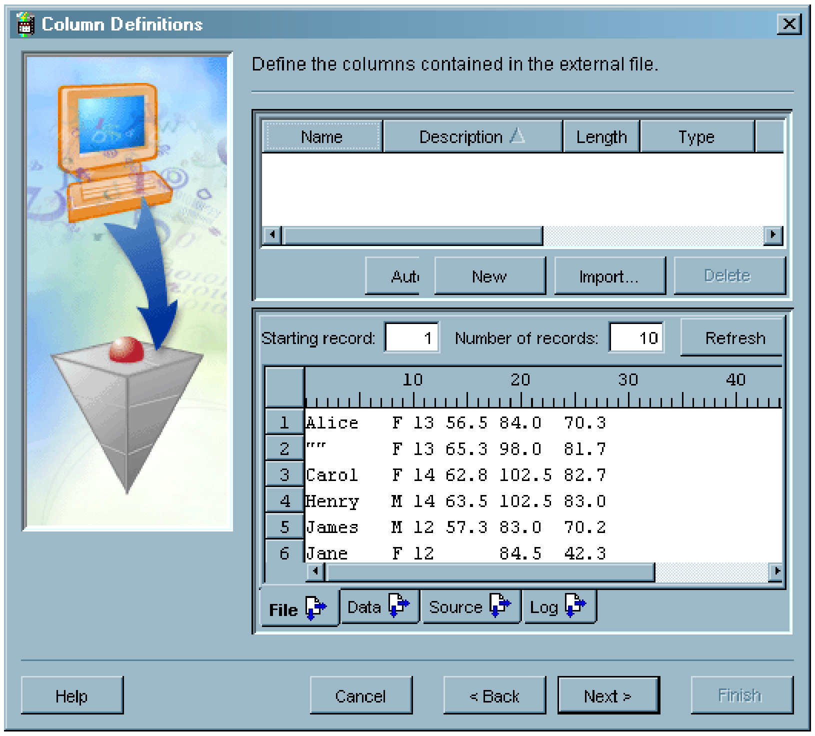
3 Specify the physical path to the external file in the **`File name`** field. Click **`Next`** to access the Parameters window.

4 The **`Pad column values with blanks`** check box is selected by default. Deselect this check box if the columns in your external file are short. It is unnecessary to pad values in short columns, and padded values can hurt performance. In addition, select the **`Treat unassigned values as missing`** check box. This setting adds the TRUNCOVER option to the SAS code, which sets variables without assigned values to missing.

5 Accept the default for the **`Logical record length`**, and click the **`Next`** button to access the Column Definitions window.

6 Click **`Refresh`** to view the raw data from the external file in the **`File`** tab in the view pane at the bottom of the window. Sample data is shown in the following display.

Display 6.5 Fixed-Width Data in the File Tab



Define the columns contained in the external file.

Starting record: 1 Number of records: 10

```
         10        20        30        40
1 Alice    F 13 56.5 84.0   70.3
2 '''      F 13 65.3 98.0   81.7
3 Carol    F 14 62.8 102.5 82.7
4 Henry    M 14 63.5 102.5 83.0
5 James    M 12 57.3 83.0   70.2
6 Jane     F 12      84.5   42.3
```

7 The next step sets the boundaries of the columns in the external file. The process is similar to the process that is used to set tabs in word processing programs. Click the appropriate tick marks in the ruler displayed at the top of the view pane. You can get the appropriate tick mark position numbers from the documentation that comes with the data. To set the first column boundary, click the tick mark on the ruler that immediately follows the end of its data. A break line displays, and the column is highlighted. For example, if the data in the first column extends to the eighth tick mark, you should click the ninth mark. Notice that the metadata for the column is also populated into the column component at the top of the window.

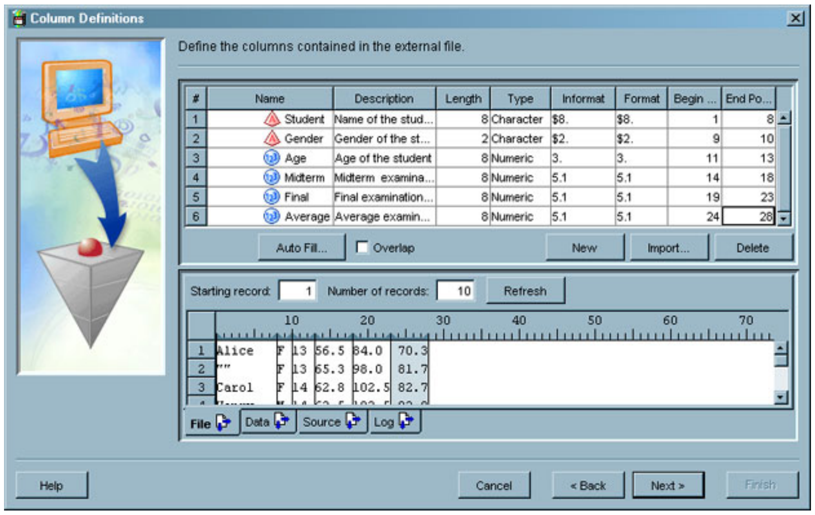
8 Click the appropriate tick marks in the ruler for the other columns in the external file. Break lines and metadata for these columns are set.

9 Click **`Auto Fill`** to refine this preliminary data by using the auto fill function. Accept all default settings and then click **`OK`** to return to the Column Definitions window. More accurate metadata is entered into the column components section of the window.

10 The preliminary metadata that is populated into the columns component usually includes column names and descriptions that are too generic to be useful for SAS Data Integration Studio jobs. Fortunately, you can modify the columns component by clicking in the cells that you need to change and by entering the correct data.

Note: The only values that need to be entered for the sample file are appropriate names and descriptions for the columns in the table. The other values were created automatically when you defined the columns and clicked **`Auto Fill`**. However, you should make sure that all variables have informats that describe the data that you are importing because the auto fill function provides a best guess at the data. You need to go in and verify this guess. If appropriate informats are not provided for all variables in the fixed-width file, incorrect results can be encountered when the external file is used in a job or its data is viewed. A sample of a completed Column Definitions window is shown in the following display. △

Display 6.6 Completed Column Definitions Window



Define the columns contained in the external file.

#	Name	Description	Length	Type	Informat	Format	Begin ...	End Po...
1	Student	Name of the stud...	8	Character	$8.	$8.	1	8
2	Gender	Gender of the st...	2	Character	$2.	$2.	9	10
3	Age	Age of the student	8	Numeric	3.	3.	11	13
4	Midterm	Midterm examina...	8	Numeric	5.1	5.1	14	18
5	Final	Final examination...	8	Numeric	5.1	5.1	19	23
6	Average	Average examin...	8	Numeric	5.1	5.1	24	28

Starting record: 1 Number of records: 10

```
1  Alice   F 13 56.5 84.0  70.3
2  ""      F 13 65.3 98.0  81.7
3  Carol   F 14 62.8 102.5 82.7
```

You can click **`Data`** to see a formatted view of the external file data. To view the code that will be generated for the external file, click the **`Source`** tab. To view the SAS log for the generated code, click the **`Log`** tab. The code that is displayed in the **`Source`** tab is the code that will be generated for the current external file when it is included in a SAS Data Integration Studio job.

11 Click **`Next`** to access the Select Folder window. Select an appropriate Custom tree folder for the external file metadata.

12 Click **`Next`** to access the General window. Enter an appropriate name for the external file metadata. (If you leave the **`Name`** field blank, the file is identified by its path.) You can also enter an optional description.

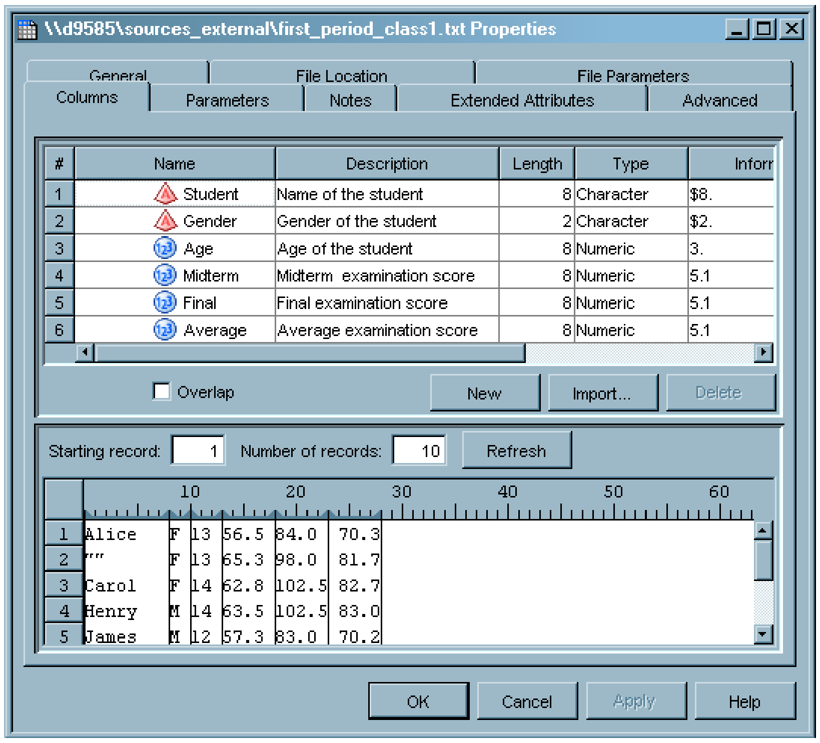
13 Click **`Finish`** to save the metadata and exit the fixed-width external file source designer.

View the External File Metadata

After you have generated the metadata for an external file, you can use SAS Data Integration Studio to view, and possibly make changes to, that metadata. For example, you might want to remove a column from a table or change the data type of a column. Any changes that you make to this metadata will not affect the physical data in the external file. However, the changes will affect the data that is surfaced when the external table is used in SAS Data Integration Studio. Perform the following steps to view or update external file metadata:

1 Right-click the external file, and click **`Properties`**. Then, click the **`Columns`** tab. The **`Columns`** tab is displayed, as shown in the example in the following display.

Display 6.7 External File Columns Tab



\\d9585\sources_external\first_period_class1.txt Properties

#	Name	Description	Length	Type	Inforr
1	Student	Name of the student	8	Character	$8.
2	Gender	Gender of the student	2	Character	$2.
3	Age	Age of the student	8	Numeric	3.
4	Midterm	Midterm examination score	8	Numeric	5.1
5	Final	Final examination score	8	Numeric	5.1
6	Average	Average examination score	8	Numeric	5.1

☐ Overlap

Starting record: 1 Number of records: 10

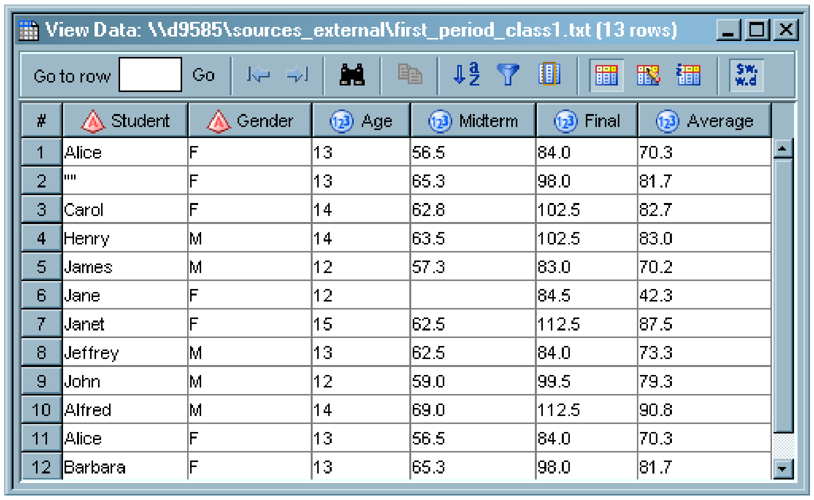
```
           10        20        30        40        50        60
1  Alice    F 13 56.5 84.0  70.3
2  ""       F 13 65.3 98.0  81.7
3  Carol    F 14 62.8 102.5 82.7
4  Henry    M 14 63.5 102.5 83.0
5  James    M 12 57.3 83.0  70.2
```

2 Click **`OK`** to save any changes and close the properties window.

View the Data

Right-click the external file, and click **`View Data`**. The View Data window is displayed, as shown in the example in the following display.

Display 6.8 External File Data in the View Data Window


View Data: \\d9585\sources_external\first_period_class1.txt (13 rows)
Go to row
Go

#	Student	Gender	Age	Midterm	Final	Average
1	Alice	F	13	56.5	84.0	70.3
2	""	F	13	65.3	98.0	81.7
3	Carol	F	14	62.8	102.5	82.7
4	Henry	M	14	63.5	102.5	83.0
5	James	M	12	57.3	83.0	70.2
6	Jane	F	12		84.5	42.3
7	Janet	F	15	62.5	112.5	87.5
8	Jeffrey	M	13	62.5	84.0	73.3
9	John	M	12	59.0	99.5	79.3
10	Alfred	M	14	69.0	112.5	90.8
11	Alice	F	13	56.5	84.0	70.3
12	Barbara	F	13	65.3	98.0	81.7

If the data in the external file displays correctly, the metadata for the file is correct and the table is available for use in SAS Data Integration Studio. If you need to review the original data for the file, right-click on its metadata object. Then, click **`View File`**.

Additional Information

General information about registering external files window is available in the "Managing External Files in SAS Data Integration Studio" Help topic in SAS Data Integration Studio. For detailed information about specific usage issues, see the "Usage Notes for External Files" Help topic.

Registering an External File with User-Written Code

Problem

You want to use user-written code to create metadata for an external file so that it can be used in SAS Data Integration Studio.

Solution

Use the user-written external file source designer to register external files with user-written SAS code that includes an INFILE statement. The metadata is saved to a SAS Metadata Repository. This metadata can then be used as a source or a target in a SAS Data Integration Studio job.

Tasks

Test Your Code

You should test your SAS code before you run it in the User Written External File source designer. That way, you can ensure that any problems that you encounter in the wizard come from the wizard itself and not from the code. Perform the following steps to run this test:

1. Open the Source Editor from the **Tools** menu in the menu bar on the SAS Data Integration Studio desktop.
2. Paste the SAS code into the Source Editor window. Here is the code that is used in this example:

   ```
   libname temp base '\\d9585\output_sas';
   %let _output=temp.temp;
   data &_output;

      infile '\\d9585\sources_external\birthday_event_data.txt'
             lrecl = 256
             pad
             firstobs = 2;

      attrib Birthday length = 8   format = ddmmyy10.  informat = YYMMDD8. ;
      attrib Event    length = $19 format = $19.       informat = $19.     ;
      attrib Amount   length = 8   format = dollar10.2 informat = comma8.2 ;
      attrib GrossAmt length = 8   format = Dollar12.2 informat = Comma12.2;

      input  @ 1  Birthday YYMMDD8.
             @ 9  Event    $19.
             @ 28 Amount   Comma8.2
             @ 36 GrossAmt Comma12.2;

   run;
   ```

   Note: The first two lines of this SAS code are entered to set the LIBNAME and output parameters that the SAS code needs to process the external file. After you have verified that the code ran successfully, delete them. They are not needed when the SAS code is used to process the external file. △

3. Review the log in the Source Editor window to ensure that the code ran without errors. The expected number of records, variables, and observations should have been created.
4. Close the Source Editor window. Do not save the results.

Run the User-Written External File Source Designer

Perform the following steps to use one method to register an external file in the user-written source designer:

1. Open the Source Designer window. Open the **External Files** folder and click **User Written External File**. Click **Next** to access the User Written Source Code window.
2. If you are prompted, enter the user ID and password for the default SAS Application Server that is used to access the external file.

3 Enter the appropriate value in the **`Type`** field. The available types are **`file`** and **`metadata`**.

4 Verify that the correct server is displayed in the **`Host`** field.

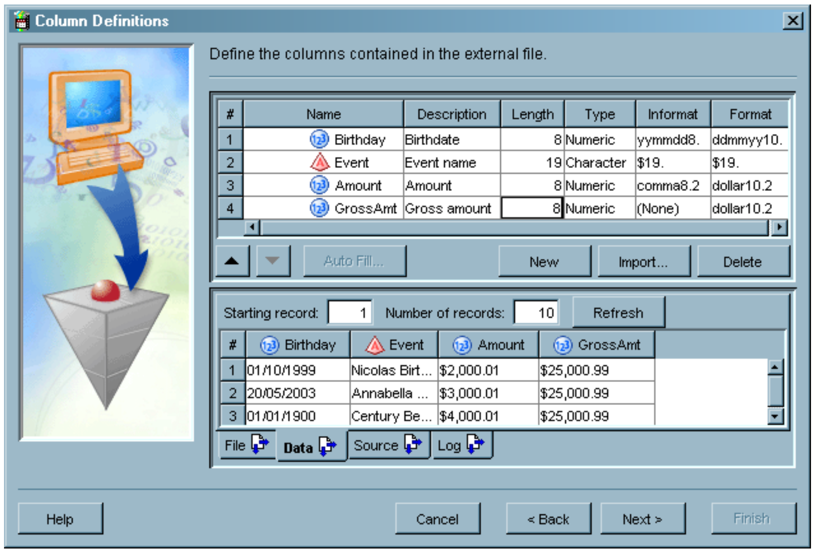
5 Specify the physical path to the external file in the **`Path`** field. Click **`Next`** to access the Column Definitions window.

6 You can either enter the column definitions manually or click **`Import`** to access the Import Column Definitions window. For information on the column import functions available there, see the "Import Column Definitions Window" in the SAS Data Integration Studio help. The column definitions for this example were entered manually.

You can find the information that you need to define the columns in the attributes list in the SAS code file. For example, the first variable in the birthday_event_code.sas file has a name of **`Birthday`**, a length of **`8`**, the **`yymmdd8.`** informat, and the **`ddmmyy10.`** format. Click **`New`** to add a row to the columns component at the top of the Column Definitions window.

7 Review the data after you have defined all of the columns. To view this data, click the **`Data`** tab under the view pane at the bottom of the window. To view the code that will be generated for the external file, click the **`Source`** tab. To view the SAS log for the generated code, click the **`Log`** tab. The code that is displayed in the **`Source`** tab is the code that will be generated for the current external file when it is included in a SAS Data Integration Studio job. The following display shows the completed Column Definitions window.

Display 6.9 Completed Column Definitions Window



Define the columns contained in the external file.

#	Name	Description	Length	Type	Informat	Format
1	Birthday	Birthdate	8	Numeric	yymmdd8.	ddmmyy10.
2	Event	Event name	19	Character	$19.	$19.
3	Amount	Amount	8	Numeric	comma8.2	dollar10.2
4	GrossAmt	Gross amount	8	Numeric	(None)	dollar10.2

Starting record: 1 Number of records: 10

#	Birthday	Event	Amount	GrossAmt
1	01/10/1999	Nicolas Birt...	$2,000.01	$25,000.99
2	20/05/2003	Annabella ...	$3,000.01	$25,000.99
3	01/01/1900	Century Be...	$4,000.01	$25,000.99

8 Click **`Next`** to access the Select Folder window. Select an appropriate Custom tree folder for the external file metadata.

9 Click **`Next`** to access the General window. Enter an appropriate name for the external file metadata. (If you leave the **`Name`** field blank, the file is identified by its path.) You can also enter an optional description.

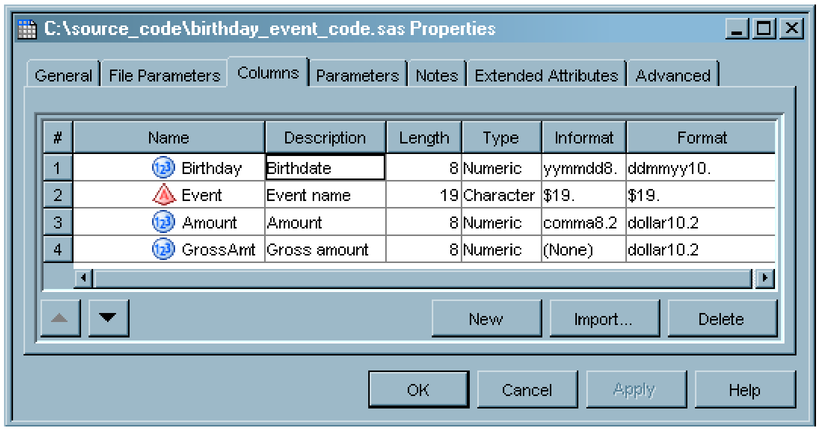
10 Click **`Finish`** to save the metadata and exit the fixed-width external file source designer.

View the External File Metadata

After you have generated the metadata for an external file, you can use SAS Data Integration Studio to view, and possibly make changes to, that metadata. For example, you might want to remove a column from a table or change the data type of a column. Any changes that you make to this metadata will not affect the physical data in the external file. However, the changes will affect the data that is included when the external table is used in SAS Data Integration Studio. Perform the following steps to view or update external file metadata:

1 Right-click the external file, and click **`Properties`**. Then, click the **`Columns`** tab. The **`Columns`** tab is displayed, as shown in the example in the following display.

Display 6.10 External File Columns Tab



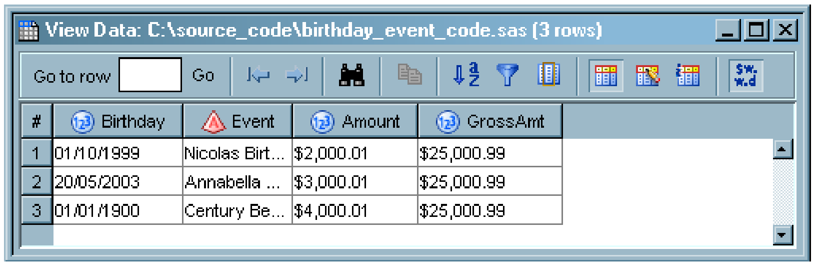
#	Name	Description	Length	Type	Informat	Format
1	Birthday	Birthdate	8	Numeric	yymmdd8.	ddmmyy10.
2	Event	Event name	19	Character	$19.	$19.
3	Amount	Amount	8	Numeric	comma8.2	dollar10.2
4	GrossAmt	Gross amount	8	Numeric	(None)	dollar10.2

2 Click **`OK`** to save any changes and close the properties window.

View the Data

Right-click the external file, and click **`View Data`**. The View Data window is displayed, as shown in the example depicted in the following display.

Display 6.11 External File Data in the View Data Window



#	Birthday	Event	Amount	GrossAmt
1	01/10/1999	Nicolas Birt...	$2,000.01	$25,000.99
2	20/05/2003	Annabella ...	$3,000.01	$25,000.99
3	01/01/1900	Century Be...	$4,000.01	$25,000.99

If the data in the external file displays correctly, the metadata for the file is correct and the table is available for use in SAS Data Integration Studio. If you need to review the original data for the file, right-click on its metadata object. Then, click **`View File`**.

Additional Information

General information about registering external files window is available in the "Managing External Files in SAS Data Integration Studio" Help topic in SAS Data

Integration Studio. For detailed information about specific usage issues, see the "Usage Notes for External Files" Help topic.

Viewing or Updating External File Metadata

Problem

You want to view or update the metadata for an external file that you have registered in SAS Data Integration Studio.

Solution

You can access the properties window for the table and change the settings on the appropriate tab of the window. The following tabs are available on properties windows for tables:

- General
- File Location (not available for user-written external files)
- File Parameters
- Columns
- Parameters
- Notes
- Extended Attributes
- Advanced

Use the properties window for an external file to view or update the metadata for its columns, file locations, file parameters, and other attributes. You can right-click an external file in any of the trees on the SAS Data Integration Studio desktop or in the Process Designer window. Then, click **`Properties`** to access its properties window.

Note that any updates that you make to an external file change the physical external file when you run a job that contains the file. These changes can have the following consequences for any jobs that use the external file:

- Changes, additions, or deletions to column metadata are reflected in all of the jobs that include the external file.
- Changes to column metadata often affect mappings. Therefore, you might need to remap your columns.
- Changes to file locations, file parameters, and parameters affect the physical external file and are reflected in any job that the includes the external file.

You can use the impact analysis and reverse impact tools in SAS Data Integration Studio to estimate the impact of these updates on your existing jobs. For information, see “About Impact Analysis and Reverse Impact Analysis” on page 255.

Overriding the Code Generated by the External File Wizards

Problem

You want to substitute your own SAS INFILE statement for the code that is generated by the Delimited External File wizard and the Fixed Width External File wizard. For details about the SAS INFILE statement, see *SAS Language Reference: Dictionary*.

Solution

Use the **`Override generated INFILE statement with the following statement`** check box in the Advanced File Parameters window of the Source Designer wizard.

Note: If you override the generated code that is provided by the external file wizards and specify a non-standard access method such as PIPE, FTP, or a URL, then the **`Preview`** button on the External File Location window, the **`File`** tab on the Columns Definition window, and the **`Auto Fill`** button on the Columns Definition window will not work. △

Tasks

Replace a Generated SAS INFILE Statement

Perform the following steps to substitute your own SAS INFILE statement for the code that is generated by the Delimited External File wizard and the Fixed Width External File wizard.

1 Open the Source Designer selection window, and select either the Delimited External File wizard or the Fixed Width External File wizard.

2 Specify the physical path for the external file and click **`Next`**. Either the Parameters window or the Parameters/Delimiters window displays (depending on the selected wizard.

3 Click the **`Advanced`** button to display the Advanced File Parameters window.

4 Select the **`Override generated INFILE statement with the following statement`** check box. Then, paste your SAS INFILE statement into the text area.

5 Enter other metadata for the external file as prompted by the wizard.

Additional Information

For details about the effects of using overridden code with a non-standard access method, see the "Accessing Data With Methods Other Than the SAS Application Server" topic in SAS Data Integration Studio help.

Specifying NLS Support for External Files

Problem

You want to specify the National Language Support (NLS) encoding for an external file. You must have the proper NLS encoding to view the contents of the selected file or automatically generate its column metadata.

Solution

Enter the appropriate encoding value into the **`Encoding options`** field in the Advanced File Parameters window of the Source Designer wizard.

Tasks

Specify NLS Encoding Options

Perform the following steps to specify NLS encoding for the Delimited External File wizard or the Fixed Width External File wizard.

1 Open the Source Designer selection window, and click either **`Delimited External File`** or Fixed Width External File.

2 Specify the physical path for an external file for which NLS options must be set, such as a Unicode file. Normally, after you have specified the path to the external file, you can click **`Preview`** to display the raw contents of the file. However, the **`Preview`** button will not work yet, because the required NLS options have not been specified.

3 Click **`Next`**. Either the Parameters window or the Parameters/Delimiters window displays (depending on the selected wizard.

4 Click **`Advanced`** to display the Advanced File Parameters window.

5 Enter the appropriate NLS encoding for the selected file in the **`Encoding options`** field. Then, click **`OK`**.

Additional Information

For detailed information about encoding values, see "Encoding Values in SAS Language Elements" in *SAS National Language Support (NLS): User's Guide*.

Accessing an External File With an FTP Server or an HTTP Server

Problem

You want to access an external file that is located on either an HTTP server or an FTP server. The Delimited External File wizard and the Fixed Width External File wizard prompt you to specify the physical path to an external file. By default, a SAS Application Server is used to access the file. However, you can access the file with an HTTP server, HTTPS server, or FTP server if the metadata for that server is available in a current metadata repository.

Note: If you use a method other than a SAS Application Server to access an external file, then the **`Preview`** button on the External File Location window, the **`File`** tab on the Columns Definition window, and the **`Auto Fill`** button on the Columns Definition window will not work. △

Solution

You can select the server in the **`FTP Server`** field or the **`HTTP Server`** field. These fields are located on the **`Access Method`** tab in the Advanced File Location Settings window of the Source Designer wizard.

Tasks

Select an HTTP server or an FTP server

Perform the following steps to select an HTTP server or an FTP server in the external file wizards:

1 Open the Source Designer selection window, and click either **`Delimited External File`** or **`Fixed Width External File`**.

2 Click **`Advanced`** in the External File Location window. The Advanced File Location Settings window displays.

3 Click the **`Access Method`** tab. Then, select either the **`FTP`** check box or the **`URL`** check box.

4 Select either an FTP server or an HTTP server in the **`FTP Server`** field or the **`HTTP Server`** field. Click **`OK`** to save the setting and close the Advanced File Location Settings window.

5 Specify a physical path for the external file. The path must be appropriate for the server that you selected.

6 Enter other metadata for the external file as prompted by the wizard.

Additional Information

For details about defining metadata for an HTTP server, HTTPS server, or an FTP server, administrators should see "Enabling the External File Wizards to Retrieve Files Using FTP or HTTP" in the SAS Data Integration Studio chapter of *SAS Intelligence Platform: Desktop Application Administration Guide*. Also see the usage note "Accessing Data With Methods Other Than the SAS Application Server" in the "Usage Notes for External Files" topic in SAS Data Integration Studio help.

Viewing Data in External Files

Problem

You want to view raw data or formatted data in the one of the external file wizards that are included in the Source Designer. You might also need to view this raw or formatted data in an external file that you have already registered by using of the external file wizards.

Solution

You can view raw data in the External File Location window or Columns Definition window in the external file wizards or in the View File window for a registered external file. You can view formatted data in the Columns Definition window in the external file wizards or in the View Data window for a registered external file.

Tasks

View Raw Data in an External File

You can click `Preview` on the External File Location window in the external file wizards to view raw data for an unregistered file. You can also click the `File` tab on the Columns Definition window. There are two main situations where the `Preview` button and the `File` tab will not be able to display data in the external file:

□ when you use a method other than a SAS Application Server to access the external file. (See "Accessing an External File With an FTP Server or an HTTP Server" on page 136.)

□ when you use the User Written External File wizard (because your SAS code, not the wizard, is manipulating the raw data in the file).

For an example of how you can use the `File` tab to help you define metadata, see the explanation of the Column Definitions window in "Registering a Delimited External File" on page 122. You can also view the raw data in an external file after you have registered it in the Source Designer. To do this, access the View File window for the external file. The raw data surfaced in the external file wizards and the View File window is displayed without detailed column specifications or data formatting. You can use it to understand the structure of the external file better.

View Formatted Data in the External File Wizards

The `Data` tab on the Columns Definition window displays data in the external file after metadata from the external file wizard has been applied. Use the `Data` tab to verify that the appropriate metadata has been specified for the external file.

The `Data` tab is populated as long as the SAS INFILE statement that is generated by the wizard is valid. The tab cannot display data for a fixed-width external file unless the SAS informats in the metadata are appropriate for the columns in the data. For an example of how you can use the `Data` tab to help you verify your metadata, see the explanation of the Column Definitions window in "Registering a Delimited External File" on page 122.

You can also view the formatted data in an external file after you have registered it in the Source Designer. To do this, access the View Data window for the external file.

Registering a COBOL Data File That Uses a COBOL Copybook

Problem

You want to create metadata for a COBOL data file that uses column definitions from a COBOL copybook. The copybook is a separate file that describes the structure of the data file.

Solution

Perform the following steps to specify metadata for a COBOL data file in SAS Data Integration Studio:

1. Use the import COBOL copybook feature to create a COBOL format file from the COBOL copybook file.
2. Use the Fixed-Width External File wizard to copy column metadata from the COBOL format file.

Tasks

Import the COBOL Copybook

Server administrators should perform the following steps, which describe one way to import the COBOL copybook:

1. Obtain the required set of SAS programs that supports copybook import. Perform the following steps from Technical Support document TS-536 to download the version of COB2SAS8.SAS that was modified for SAS V8:
   a. Go to the following Web page and download this zipped file: **`http://ftp.sas.com/techsup/download/mvs/cob2sas8.zip`**.
   b. Unzip the file into an appropriate directory.
   c. Read the README.TXT file. It contains information about this modified version of COB2SAS8.SAS. It also contains additional information about the installation process.
2. Click **`Import COBOL Copybook`** in the **`Tools`** menu for SAS Data Integration Studio to access the Cobol Copybook Location and Options window.
3. Select a SAS Application Server in the **`Server`** field. The selected SAS Application Server must be able to resolve the paths that are specified in the **`Copybook(s)`** field and the **`COBOL format file directory`** field.
4. Indicate the original platform for the COBOL data file by selecting the appropriate radio button in the **`COBOL data resides on`** field.

5 Select a copybook file to import in the **Copybook(s)** field. If you have imported copybooks in the past, you can select from a list of up to eight physical paths to previously selected copybook files. If you need to import a copybook that you have never used in SAS Data Integration Studio, you have two options. First, you can click **Add** to type a local or remote path manually. Second, you can click **Browse** to browse for a copybook that is local to the selected SAS Application Server.

6 Specify a physical path to the directory for storing the COBOL format file in the **COBOL format file directory** field. You can enter a local or remote path in the field, choose a previously selected location from the drop-down menu, or browse to the file.

7 Click **OK** when you are finished. The Review object names to be created window displays.

8 Verify the name of the COBOL format file or files. Specify a physical path for the SAS log file in the **SAS Log** field. This file will be saved to the SAS Data Integration Studio client machine.

9 Click **OK** when you are finished. One or more COBOL format files are created from the COBOL copybook file.

Note: If the external file resides on the MVS operating system, and the filesystem is native MVS, then the following usage notes apply. △

- □ Add the **MVS:** tag as a prefix to the name of the COBOL copybook file in the **Copybook(s)** field . Here is an example file name: **MVS:wky.tst.v913.etls.copybook**.
- □ Native MVS includes partitioned data sets (PDS and PDSE). Take this into account when you specify a physical path to the directory for storing the COBOL format file in the **COBOL format file directory** field . Here is an example path: **MVS:dwatest.tst.v913.cffd**.
- □ The COB2SAS programs must reside in a PDS with certain characteristics. For more information about these characteristics, see **http://support.sas.com/techsup/technote/ts536.html**.
- □ The path to the **r2cob1.sas** program should specify the PDS and member name. Here is an example path, which would be specified in the **Full path for r2cob1.sas** field in the Advanced options window: **mvs:dwatest.tst.v913.cob2sasp(r2cob1)**.

Copy Column Metadata From the COBOL Format File

You can copy column metadata from the COBOL format file in the Column Definitions window of the Fixed Width External File wizard. Perform the following steps:

1 Access the Column Definitions screen of the Fixed Width External File wizard. For information about the wizard, see "Registering a Fixed-Width External File" on page 126.

2 Click **Import** to access the Import Columns window.

3 Select the **Get the column definitions from a COBOL copybook** radio button. Then, use the down arrow to select the appropriate COBOL format file and click **OK**. The column metadata from the COBOL format file is copied into the Column Definitions window.

4 Specify any remaining column metadata in the Column Definitions window. Click **Next** when you are finished.

5 Click through the Select Group and General windows of the wizard, entering any unfinished configuration as you go. Click **Finish** when you are finished. The metadata for the external file is saved into the appropriate repository.

CHAPTER 7 Creating, Executing, and Updating Jobs

About Jobs

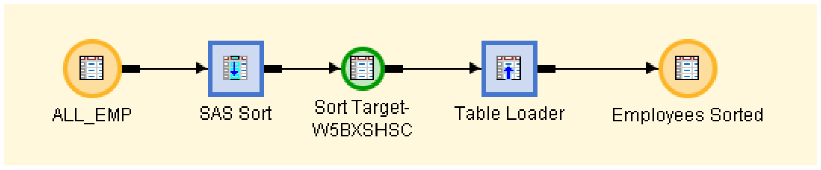
Jobs with Generated Source Code

A job is a collection of SAS tasks that create output. SAS Data Integration Studio uses the metadata for each job to generate or retrieve SAS code that reads sources and creates targets in physical storage.

If you want SAS Data Integration Studio to generate code for a job, you must define a process flow diagram that specifies the sequence of each source, target, and process in a job. In the diagram, each source, target, and process has its own metadata object.

For example, the following process flow diagram shows a job that reads data from a source table, sorts the data, and then writes the sorted data to a target table.

Display 7.1 Process Flow Diagram for a Job That Sorts Data



Given the direction of the arrows in the process flow:

- ALL_EMP specifies metadata for the source table.
- SAS Sort specifies metadata for the sort process, which writes its output to a temporary output table, Sort Target-W5BU8XGB.
- Table Loader specifies metadata for a process that reads the output from the previous step and loads this data into a target table.
- Employees Sorted specifies metadata for the target table.

SAS Data Integration Studio uses this metadata to generate SAS code that reads ALL_EMP, sorts this information, writes the sorted information to a temporary output table, and then writes it to the Employees Sorted table.

Each process in a process flow diagram is specified by a metadata object called a transformation. In the example, SAS Sort and Table Loader are transformations. A transformation specifies how to extract data, transform data, or load data into data stores. Each transformation that you specify in a process flow diagram generates or retrieves SAS code. You can specify user-written code for any transformation in a process flow diagram.

For more details about the process flow diagram shown in the preceding example above, see "Creating a Table That Sorts the Contents of a Source" on page 279.

Jobs with User-Supplied Source Code

For all jobs except the read-only jobs that create cubes, you can specify user-written code for the entire job or for any transformation within the job. For details, see Chapter 12, "Working with User-Written Code," on page 215.

Run Jobs

There are four ways to run a job:

- submit the job for immediate execution; see "Submitting a Job for Immediate Execution" on page 147
- deploy the job for scheduling; see "About Job Scheduling" on page 170
- deploy the job as a SAS stored process; see "About SAS Stored Processes" on page 177
- deploy the job as a SAS stored process that can be accessed by a Web service client; see "About Deploying Jobs for Execution by a Web Service Client" on page 182

Manage Job Status

After you have submitted the job, you can review and manage its status in the Job Status Manager window. The Job Status Manager displays all of the jobs that have been submitted and not cleared in the current SAS Data Integration Studio session. You can use this tool to view, clear, cancel, kill, and resubmit jobs. For details, see "Using the Job Status Manager" on page 159.

Creating an Empty Job

Problem

You want to create an empty job. After you have an empty job, you can create a process flow diagram by dragging and dropping tables and transformations into the Process Designer window.

Solution

Use the New Job wizard.

Tasks

Use the New Job Wizard

Perform the following steps to create an empty job:

1 Access the New Job wizard through one of the following methods:
   - □ Click **`Process Designer`** on the shortcuts bar.
   - □ Select **Tools** ▶ **Process Designer**.
   - □ Select **File** ▶ **New Object** Then, click **`New Job`** on the New Object Wizard window.

2 Enter an appropriate name for the job in the New Job wizard. You can also enter an optional description of the job.

3 Click **`Next`**. Do not select tables to be loaded in this empty job. (You can use this window to select target tables for a job.)

4 Click **`Next`**. Select an appropriate group in the SAS Data Integration Studio Custom Tree.

5 Click **`Next`**, review the metadata for the job, and click **`Finish`**. The Process Designer displays an empty job.

Related Tasks

Save the new job before you close the Process Designer window. To save the job, select **File** ▶ **Save** from the menu bar.

After you have created an empty job, you can populate and execute the job.

Creating a Process Flow for a Job

Problem

You want to create a job to perform a business task. Then, you need to populate the job with the source tables, transformations, and target tables required to complete the task.

Solution

You can use the New Job Wizard to create an empty job. Then, you can populate the job in the Process Designer window with the source tables, transformations, and target tables that you need to accomplish your task.

Tasks

Create and Populate a Sample Job

To illustrate the process of creating and populating a job, we will build a sample job. Perform the following steps to create and populate the job:

1 Create an empty job. For information, see “Creating an Empty Job” on page 143.

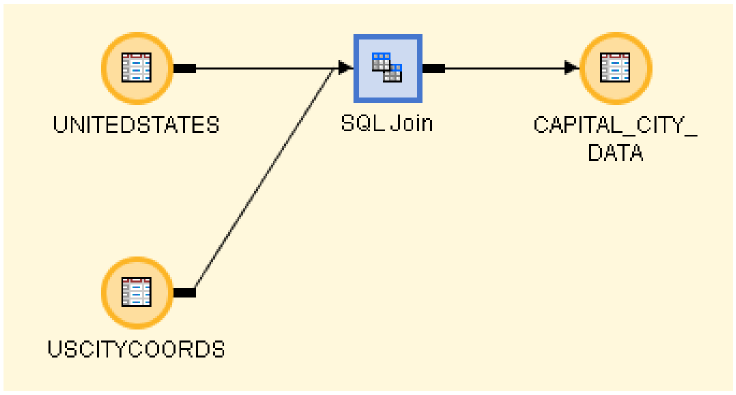
2 Drop the SQL Join transformation from the **`Data Transformations`** folder in the Process Library tree into the empty job. You generally drop the transformation into the flow first. This way, the appropriate input and output drop zones are provided.

3 Drop the first source table on the input drop zone for the SQL Join transformation. Then, drop any additional source tables on the SQL Join transformation. The source tables can be tables or external files. However, they must be registered in SAS Data Integration Studio.

4 Drop the target table on the target drop zone. The target table must be registered in SAS Data Integration Studio.

5 Delete the Table Loader transformation and the temporary worktable SQL Target from the job. If you keep the Table Loader and the worktable, you must configure two sets of mappings, one from the source tables to the worktable and another from the worktable to the target table. The extra processing required can degrade performance when the job is run. For information that will help you decide whether to delete the Table Loader transformation and the temporary worktable from a job, see “Manage Temporary and Permanent Tables for Transformations” on page 239. The following example shows the sample process flow.

Display 7.2 Sample Process Flow



Note: You can set global options for jobs on the **`Code Generation`** tab of the **`Options`** menu. The **`Options`** menu is available on the **`Tools`** menu on the SAS Data Integration Studio menu bar. You can set local options on the **`Options`** tab available on the properties window for each table. For detailed information, see “Specifying Options for Jobs” on page 211. △

About Job Options

Options can be set for Data Integration Studio, such as enabling parallel processing and configuring grid processing.

Use the Options window to specify options for SAS Data Integration Studio. You can display this window by selecting **Tools** ▶ **Options** from the menu bar.

In most cases the appropriate options are selected by default. You can override the defaults by using one of the options in the following tables.

Table 7.1 Global Options for Jobs

Option name	Description
Enable optional macro variables for new jobs	When selected, specifies that optional macro variables are to be included in the code that SAS Data Integration Studio generates for new jobs.
Enable parallel processing macros for new jobs	When selected, adds parallel processing macros to the code that is generated for all new jobs.
Default grid workload specification	Enables you to select a default workload specification value for all new jobs. For example, if the grid is partitioned, you can designate that specific applications will run on designated servers. The grid workload specification consists of a string value that must match the name defined in the Platform Computing grid configuration files. These files are text files set up by administrators when they configure a grid.
Default maximum number of concurrent processes group box	Contains concurrent processes options.
One process for each available CPU node	When selected, sets the number of concurrent processes to one process for each available CPU node for all new jobs. Generally, this is the most effective setting.
Use this number	Specifies an exact number of concurrent processes to run for all new jobs.
Run all processes concurrently	When selected, runs all processes concurrently by using SAS load balancing for new jobs. Typically, this option is used only in a grid computing environment where a managing scheduler, such as Platform Computing software, is used to handle workload management for the grid. In a grid computing environment, you should also adjust your job slots to match your environment and perform other necessary tuning. Too many processes sent to an overloaded environment can dramatically reduce performance, and potentially cause deadlock.

You can set local options that apply to individual jobs by selecting the job and using the right mouse button to open the pop-up menu. Select **`Properties`** and then select the **`Options`** tab. These local options override global options for the selected job, but they do not affect any other jobs.

Table 7.2 Local Options for Jobs

Option name	Description
Enable optional macro variables	When set to YES, specifies that optional macro variables are to be included in the code that SAS Data Integration Studio generates for the selected job. This option overrides the global option with the same name.
Enable parallel processing macros	When set to YES, adds parallel processing macros to the code that is generated for the selected job. This option overrides the global option with the same name.
System Options	Enables you to set options by using a SAS OPTIONS statement.

Submitting a Job for Immediate Execution

Problem

You want to execute a job immediately.

Solution

You can submit jobs from the Process Designer window, from any tree on the SAS Data Integration Studio desktop, or from the Job Status Manager window.

You can submit a job after you have defined its metadata. Until you submit a job, its output tables (or targets) might not exist on the file system. Note that you can open multiple jobs in multiple Process Designer windows and submit each job for execution. These jobs execute in the background, so you can do other tasks in SAS Data Integration Studio while a job is executing. Each job has its own connection to the SAS Application Server so that the jobs can execute in parallel.

Note: Two jobs that load the same target table should not be executed in parallel. They will either overwrite each other's changes, or they will try to open the target at the same time. △

The SAS Application Server that executes the job must have been installed, and the appropriate metadata must have been defined for it. For details, see "Selecting a Default SAS Application Server" on page 55.

Tasks

Submit a Job from a Process Designer Window

You can submit a job that is displayed in a Process Designer window. Click **`Submit`** in the SAS Data Integration Studio **`Process`** menu. The job is submitted to the default

SAS Application Server and to any server that is specified in the metadata for a transformation within the job.

Submit a Job from a Tree

You select a job in the Custom tree, the Metadata tree, or the Inventory tree. After the job is selected, you can view it in the Process Designer window, submit it, or review its code. Perform the following steps if the job to be submitted is not displayed in the Process Editor:

1 Display the tree in the SAS Data Integration Studio desktop where the job is stored.
2 Navigate to the job in the selected tree. For example, jobs are located in the **`Jobs`** folder of the Inventory tree.
3 Right-click the job. Then, click **`Submit Job`** in the pop-up menu. If you need to review the job before you submit it, perform the following steps:
   - □ Right-click the job that you want to execute. Then, click **`View Job`** in the pop-up menu. You can also access **`View Job`** from the **`View`** menu in the menu bar. The process flow diagram for the job displays in the Process Designer.
   - □ Right-click in the Process Designer. Then, click **`Submit`** in the pop-up menu.

Resubmit a Job from the Job Status Manager Window

You can resubmit any job that is listed in the Job Status Manager window. Right-click on the row for the job in the Job Status Manager window and click **`Resubmit`** from the pop-up menu.
The job is submitted to the default SAS Application Server and to any server that is specified in the metadata for a transformation within the job.

Note: After the job is submitted, you can review and manage its status in the Job Status Manager window. The Job Status Manager window displays all jobs that have been submitted and not cleared in the current SAS Data Integration Studio session. For details, see "Using the Job Status Manager" on page 159. △

If the job executes without error, you can view the data in one or more the targets to verify that they have the appropriate content. For details, see "Browsing Table Data" on page 109 .
If the job fails to execute without error, see "Troubleshooting a Job" on page 155.

Accessing Local and Remote Data

Access Data in the Context of a Job

You can access data implicitly in the context of a job, access data interactively in a job, or use the Data Transfer transformation to move data directly from one machine to another.
When code is generated for a job, it is generated in the current context. The context includes the default SAS Application Server when the code was generated, the credentials of the person who generated the code, and other information. The context of a job affects the way that data is accessed when the job is executed.

In order to access data in the context of a job, you need to understand the distinction between local data and remote data. Local data is addressable by the SAS Application Server when code is generated for the job. Remote data is not addressable by the SAS Application Server when code is generated for the job.

For example, the following data is considered local in the context of a job:

- data that can be accessed as if it were on one or more of the same computers as the SAS Workspace Server components of the default SAS Application Server
- data that is accessed with a SAS/ACCESS engine (used by the default SAS Application Server)

The following data is considered remote in a SAS Data Integration Studio job:

- data that cannot be accessed as if it were on one or more of the same computers as the SAS Workspace Server components of the default SAS Application Server
- data that exists in a different operating environment from the SAS Workspace Server components of the default SAS Application Server (such as MVS data that is accessed by servers running under Microsoft Windows)

Note: Avoid or minimize remote data access in the context of a SAS Data Integration Studio job. △

Remote data has to be moved because it is not addressable by the relevant components in the default SAS Application Server at the time that the code was generated. SAS Data Integration Studio uses SAS/CONNECT and the UPLOAD and DOWNLOAD procedures to move data. Accordingly, it can take longer to access remote data than local data, especially for large data sets. It is especially important to understand where the data is located when using advanced techniques such as parallel processing because the UPLOAD and DOWNLOAD procedures would run in each iteration of the parallel process.

For information about accessing remote data in the context of a job, administrators should see "Multi-Tier Environments" in the SAS Data Integration Studio chapter of the *SAS Intelligence Platform: Desktop Application Administration Guide*. Administrators should also see “Deploying Jobs for Execution on a Remote Host” on page 175. For details about the code that is generated for local and remote jobs, see the subheadings about LIBNAME statements and remote connection statements in “Common Code Generated for a Job” on page 152.

Access Data Interactively

When you use SAS Data Integration Studio to access information interactively, the server that is used to access the resource must be able to resolve the physical path to the resource. The path can be a local path or a remote path, but the relevant server must be able to resolve the path. The relevant server is the default SAS Application Server, a server that has been selected, or a server that is specified in the metadata for the resource.

For example, in the source designers for external files, the **`Server`** tab in the Advanced File Location Settings window enables you to specify the SAS Application Server that is used to access the external file. This server must be able to resolve the physical path that you specify for the external file.

As another example, assume that you use the **`View Data`** option to view the contents of a table in the Inventory tree. If you want to display the contents of the table, the default SAS Application Server or a SAS Application Server that is specified in the library metadata for the table must be able to resolve the path to the table.

In order for the relevant server to resolve the path to a table in a SAS library, one of the following conditions must be met:

- The metadata for the library does not include an assignment to a SAS Application Server, and the default SAS Application Server can resolve the physical path that is specified for this library.
- The metadata for the library includes an assignment to a SAS Application Server that contains a SAS Workspace Server component, and the SAS Workspace Server is accessible in the current session.
- The metadata for the library includes an assignment to a SAS Application Server, and SAS/CONNECT is installed on both the SAS Application Server and the machine where the data resides. For more information about configuring SAS/CONNECT to access data on a machine that is remote to the default SAS Application Server, administrators should see "Multi-Tier Environments" in the SAS Data Integration Studio chapter of the *SAS Intelligence Platform: Desktop Application Administration Guide*.

Note: If you select a library that is assigned to an inactive server, you receive a "Cannot connect to workspace server" error. Check to make sure that the server assigned to the library is running and is the active server. △

Use a Data Transfer Transformation

You can use the Data Transfer transformation to move data directly from one machine to another. Direct data transfer is more efficient than the default transfer mechanism.

For example, assume that you have the following items:

- a source table on machine 1
- the default SAS Application Server on machine 2
- a target table on machine 3

You can use SAS Data Integration Studio to create a process flow diagram that moves data from the source on machine 1 to the target on machine 3. By default, SAS Data Integration Studio generates code that moves the source data from machine 1 to machine 2 and then moves the data from machine 2 to machine 3. This is an implicit data transfer. For large amounts of data, this might not be the most efficient way to transfer data.

You can add a Data Transfer transformation to the process flow diagram to improve a job's efficiency. The transformation enables SAS Data Integration Studio to generate code that migrates data directly from the source machine to the target machine.

You can use the Data Transfer transformation with a SAS table or a DBMS table whose table and column names follow the standard rules for SAS Names. For an example of how you can use a Data Transfer transformation, see the "Example: Move Data Directly from One Machine to Another Machine" Help topic in SAS Data Integration Studio. Also see the "Data Transfer Will Not Work for DBMS Tables With Special Characters in Table Names" section in the "SAS Data Integration Studio Usage Notes" Help topic.

Viewing or Updating Job Metadata

Problem

You want to view or update the metadata that is associated with a job. All jobs have basic properties that are contained in metadata that is viewed from the job properties window. If you want SAS Data Integration Studio to generate code for the job, the job must also have a process flow diagram. If you supply the source code for a job, no process flow diagram is required. However, you might want to create one for documentation purposes.

Solution

You can find metadata for a job in its properties window or process flow diagram.

Tasks

View or Update Basic Job Properties

Perform the following steps to view or update the metadata that is associated with the job properties window:

1 Open the `Jobs` folder in the Inventory tree on the SAS Data Integration Studio desktop.
2 Right-click the desired job and click `Properties` to access the properties window for the job.
3 Click the appropriate tab to view or update the desired metadata.

For details about the metadata that is maintained on a particular tab, click the `Help` button on that tab. The Help topics for complex tabs often include task topics that can help you perform the main tasks that are associated with the tab.

View or Update the Job Process Flow Diagram

Perform the following steps to view or update the process flow diagram for a job:

1 Open the `Jobs` folder in the Inventory tree on the SAS Data Integration Studio desktop.
2 Right-click the desired job and click `View Job` to display the process flow diagram for the job in the Process Designer window. View or update the metadata displayed in the process flow diagram as follows:
   - To update the metadata for tables or external files in the job, see "Viewing or Updating Table Metadata" on page 89 or "Viewing or Updating External File Metadata" on page 134.
   - To update the metadata for transformations in the job, see "Viewing or Updating the Metadata for Transformations" on page 197.
   - To add a transformation to a process flow diagram, select the transformation drop it in the Process Designer window.

Note: Updates to job metadata are not reflected in the output for that job until you rerun the job. For details about running jobs, see "Submitting a Job for Immediate Execution" on page 147. △

Displaying the SAS Code for a Job

Problem

You want to display the SAS code for a job. (To edit the SAS code for a job, see "About User-Written Code" on page 216.)

Solution

You can display the SAS code for a job in the **`Source Editor`** tab of the Process Designer window or in the View Code window. In either case, SAS Data Integration Studio must be able to connect to a SAS Application Server with a SAS Workspace Server component to generate the SAS code for a job. See "Connecting to a Metadata Server" on page 51.

Tasks

View SAS Code in the Source Editor Tab

You can view the code for a job that is currently displayed in the Process Designer window. To do this, click the **`Source Editor`** tab. The job is submitted to the default SAS Application Server and to any server that is specified in the metadata for a transformation within the job. The code for the job is displayed in the **`Source Editor`** tab.

View SAS Code in the View Code Window

Perform the following steps to view the code for a job that is not displayed in the Process Designer window:

1. Expand the **`Jobs`** folder in the Inventory tree on the SAS Data Integration Studio desktop.
2. Right-click the job that you want to view, and then select **`View Code`** from the pop-up menu.

The job is submitted to the default SAS Application Server and to any server that is specified in the metadata for a transformation within the job. The job is opened in the Process Designer window, and the code for the job is displayed in the **`Source Editor`** tab.

Common Code Generated for a Job

Overview

When SAS Data Integration Studio generates code for a job, it typically generates the following items:

- □ a LIBNAME statement for each table in the job
- □ a SYSLAST macro statement at the end of each transformation in the job
- □ remote connection statements for any remote execution machine that is specified in the metadata for a transformation within a job
- □ macro variables for status handling

The generated code includes the user name and password of the person who created the job. You can set options for the code that SAS Data Integration Studio generates for jobs and transformations. For details, see "Specifying Options for Jobs" on page 211.

LIBNAME Statements

When SAS Data Integration Studio generates code for a job, a library is considered *local* or *remote* in relation to the SAS Application Server that executes the job. If the library is stored on one of the machines that is specified in the metadata for the SAS Application Server that executes the job, it is local. Otherwise, it is remote.

SAS Data Integration Studio generates the appropriate LIBNAME statements for local and remote libraries.

The following syntax is generated for a local library:

```
libname libref <"lib-specification"> <connectionOptions> <libraryOptions>
<schema=databaseSchema> <user=userID> <password=password>;
```

The following syntax is generated for a remote library:

```
options comamid=connection_type;
%let remote_session_id=host_name <host_port>;
signon remote_session_id <user=userID password=password>;
rsubmit remote_session_id;
   libname libref <engine> <<"lib-specification"> <connectionOptions>
<libraryOptions> <password=password>;
endrsubmit;

rsubmit remote_session_id;
proc download data=table_on_remote_machine out=table_on_local_machine;
run;
endrsubmit;
```

SYSLAST Macro Statements

The **Options** tab in the property window for most transformations includes a field that is named **Create SYSLAST Macro Variable**. This field specifies whether SAS Data Integration Studio generates a SYSLAST macro statement at the end of the current transformation. In general, accept the default value of **YES** for the **Create SYSLAST Macro Variable** option when the current transformation creates an output table that should be the input of the next transformation in the process flow. Otherwise, select **NO**.

When you select **YES** for a transformation, SAS Data Integration Studio adds a SYSLAST macro statement to the end of the code that is generated for the transformation. The syntax of this statement is as follows:

```
%let SYSLAST=transformation_output_table_name;
```

The value represented by

transformation_output_table_name

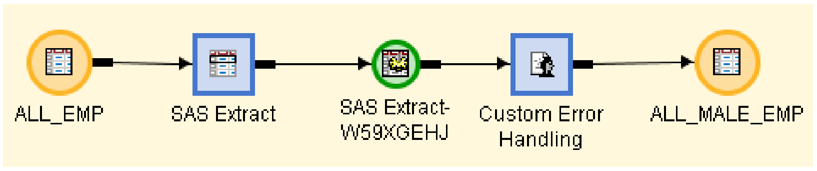
is the name of the last output table created by the transformation. The SYSLAST macro variable is used to make

transformation_output_table_name

the input for the next step in the process flow. In most cases, this setting is appropriate.

Setting the value to **NO** is appropriate when you have added a transformation to a process flow, and that transformation does not produce output, or it produces output that should not become the input to the next step in the flow. The following example illustrates a sample process flow.

Display 7.3 Process Flow with a Custom Error Handling Transformation



In this example, the Custom Error Handing transformation contains user-written code that handles errors from the Extract transformation, and the error-handling code does not produce output that should be become the input to the target table, ALL_MALE_EMP. Instead, the output from the Extract transformation should become the input to ALL_MALE_EMP.

In that case, you would do the following:

- ☐ Leave the **Create SYSLAST Macro Variable** option set to **YES** for the Extract transformation.
- ☐ Set the **Create SYSLAST Macro Variable** option to **NO** for the Custom Error Handing transformation.

Remote Connection Statements

Each transformation within a job can specify its own execution host. When SAS Data Integration Studio generates code for a job, a host is considered local or remote in relation to the SAS Application Server that executes the job. If the host is one of the machines that is specified in the metadata for the SAS Application Server that executes the job, it is local. Otherwise, it is remote.

A remote connection statement is generated if a remote machine has been specified as the execution host for a transformation within a job, as shown in the following sample statement:

```
options comamid=connection_type;
%let remote_session_id=host_name <HOST_PORT>;
SIGNON remote_session_id <USER=userID password=password>;
rsubmit remote_session_id;
  ... SAS code ...
endrsubmit;
```

Macro Variables for Status Handling

When SAS Data Integration Studio generates the code for a job, the code includes a number of macro variables that can be used to monitor the status of jobs. For details, see Chapter 8, "Monitoring Jobs," on page 157.

User Credentials in Generated Code

The code that is generated for a job contains the credentials of the person who created the job. If a person's credentials are changed and a deployed job contains outdated user credentials, the deployed job fails to execute. The solution is to redeploy the job with the appropriate credentials. For details, see "About Job Scheduling" on page 170.

Troubleshooting a Job

Problem

You have run a job that has failed to execute without error and you need to determine why the job did not execute successfully.

Solution

You can troubleshoot a job that has failed to execute without error by viewing the contents of the **Log** tab in the Process Designer window.

Tasks

Display the Log tab

Perform the following steps if the **Log** tab does not currently display in the Process Designer window:

1. Display the Options window by using one of the following methods:
   - ☐ Click the **Options** item in the Shortcut Bar.
   - ☐ Click **Options** in the **Tools** menu on the SAS Data Integration Studio menu bar.
2. Select the **Show Log tab in Process Designer** check box on the **General** tab. Then, click **OK** to save the setting.

Review the Log for a Job

Perform the following steps if a job fails to execute without error:

1. Click the **Log** tab in the Process Designer window.
2. Scroll through the SAS log information on the **Log** tab that was generated during the execution of the job. Locate the first error in the log and try to correct the problem. For example, if there are errors in the metadata for a table or transformation in the job, select it in the process flow diagram. Then, right-click it and click **Properties** in the pop-up menu to display the properties window. Correct the metadata and resubmit the job until you cannot find an error.
3. After the job runs without error, right-click anywhere in the Process Designer window and click **Save** to save the job.

Note: SAS Data Integration Studio includes a Job Status Manager that allows you to review and manage multiple jobs. For information, see "Using the Job Status Manager" on page 159. △

CHAPTER 8 Monitoring Jobs

About Monitoring Jobs

There are four ways to monitor the status of jobs and transformations in SAS Data Integration Studio:

Table 8.1 Methods for Monitoring Jobs and Transformations

If you want to	Do this
Check the status of the last five jobs that have been submitted in the current session.	Use the Job Status Manager, described in "Using the Job Status Manager" on page 159.
Define a specific action based on the return code of a job or of some transformations.	Use Status Code Handling, described in "Managing Status Handling" on page 161.

If you want to	Do this
Monitor the status of a transformation that does not have a **`Status Handling`** tab.	Insert a Return Code Check transformation into the process flow in order to check the transformation's return code. See "Managing Return Code Check Transformations" on page 162.
Monitor the status of a Lookup transformation, which has some advanced methods for handling exceptions.	See the online Help for the Lookup transformation.

When you execute a job in SAS Data Integration Studio, a return code for each transformation in the job is captured in a macro variable. The return code for the job is set according to the least successful transformation in the job.

SAS Data Integration Studio enables you to associate a return code condition, such as successful, with an action, such as **`Send Email`** or **`Send Event`**. In this way, you can specify how a return code is handled for the job or transformation. For example, you can associate a return code with an action that performs one or more of these tasks:

- Terminate the job or transformation.
- Call a user-defined SAS macro.
- Send a status message to a person, a file, or an event broker that then passes the status code to another application.

You can also use the status handling to capture job statistics, such as the number of records before and after the append of the last table loaded in the job. To capture statistics about a job, associate a return code with the Send Job Status action.

The **`Status Handling`** tab, which is included in the properties windows for jobs and for some transformations, is used to associate a return code condition with an action. See "Use the Status Handling Tab" on page 161.

The properties windows for most transformations do not have a **`Status Handling`** tab. To return the status of a transformation that does not have a **`Status Handling`** tab, you can use a Return Code Check transformation to insert status-handling logic at a desired point in the process flow diagram for a job. See "Use Return Code Check Transformations" on page 163.

Prerequisites for Job Status Code Handling

When you execute a job in SAS Data Integration Studio, notification of its success or failure can be automatically e-mailed to a person, written to a file, or sent to an event broker that passes job status on to another application. Therefore, the SAS Application Server that is used to execute jobs must have a SAS Workspace Server component that is invoked from Base SAS 9.1.2 or later. This prerequisite supports status code handling in general. The following message is displayed if you attempt to use the **`Status Handling`** tab and the current metadata repository is not set up correctly for status handling: "The metadata server does not contain the needed information to support this feature. Please notify your metadata administrator to add the default initialization metadata." If you see this message, an administrator must use the Metadata Manager plug-in to SAS Management Console to initialize the repository.

A **`Status Handling`** tab is included in the properties windows for jobs and for some transformations. Users can select from a list of code conditions and actions on the **`Status Handling`** tab. For example, you can select a code condition such as **`Successful`** and associate it with an action, such as **`Send Email`** or **`Send Event`**. The prerequisites listed in the following table apply to specific default actions.

Table 8.2 Status Handling Actions

Action	Description
Email	SAS system options for e-mail must be set for the SAS Application Server that is used to execute jobs. For details about the relevant e-mail options, see the “Administering SAS Data Integration Studio” chapter in the *SAS Intelligence Platform: Desktop Application Administration Guide*.
Send Event	SAS Foundation Services must be installed, and the Event Broker Service must be properly configured for the SAS solution that will receive the events. For details about setting up the Event Broker Service so that a SAS solution can receive events from SAS Data Integration Studio, see the documentation for the SAS solution.
Custom	An appropriate SAS Autocall macro library must be accessible by the SAS Application Server that is used to execute the job. For details about making Autocall macro libraries available to SAS Data Integration Studio, see the “Administering SAS Data Integration Studio” chapter in the *SAS Intelligence Platform: Desktop Application Administration Guide*.
Send Entry to a Data Set	The library that contains the data set must be assigned before the job or transformation is executed. To assign a library within SAS Data Integration Studio, you can select the **`Pre and Post Process`** tab in the properties window for the job or transformation and then specify a SAS LIBNAME statement as a preprocess. To assign a library outside of SAS Data Integration Studio, you can preassign the library to the SAS Application Server that is used to execute the job. Some tasks that are associated with preassigning a SAS library must be done outside of SAS Data Integration Studio or SAS Management Console. For details, see the “Assigning Libraries” chapter in *SAS Intelligence Platform: Data Administration Guide*.
Send Job Status	Saves status messages to a SAS data set. Consecutive messages are appended to the data set with a timestamp. If the data set does not exist, it is created the first time that the job is executed

Using the Job Status Manager

Problem

You have submitted one or more jobs for execution and need to review the status of the jobs. Depending on the status, you might want to clear, kill, resubmit, or perform another action on the job.

Solution

You can display a pop-up menu that shows the status of the last five unique jobs that have been submitted in the current SAS Data Integration Studio session. You can also display the Job Status Manager window, which shows the name, status, starting time, ending time, and application server used for all jobs submitted in the current session.

Tasks

Review a Job in the Job Status Manager

Perform the following steps to review a job the Job Status Manager window:

1 On the SAS Data Integration Studio desktop, select the **`Inventory`** tab.
2 In the Inventory tree, open the **`Jobs`** folder.
3 Select **Tools ▶ Job Status Manager**. A list is displayed of the last five unique jobs that have been submitted in the current SAS Data Integration Studio session and the status for each of them.
4 Right-click on a job. A pop-up menu enables you to clear, view, cancel, kill, or resubmit the job.
5 Right-click on any column heading to sort the list of jobs.

You can click **`Clear All`** to clear all of the completed jobs that are displayed in the Job Status Manager table. You can also right-click in the Job Status Manager window to access a pop-up menu that enables you to control the sort order and manage jobs. The following table explains the pop-up menu. The options that you see vary according to the context.

Table 8.3 Job Status Manager Window Pop-Up Menu Options

Menu option	Available when you	Description
Cancel Job	Right-click on a currently running job.	Stops the selected running job after the current SAS procedure or DATA step completes.
Clear	Right-click on a completed job.	Clears the selected job and disconnects the application server. Available only for completed jobs.
Clear All	Right-click on any job or in any blank space within the Job Status Manager window.	Clears all completed jobs. Disconnects the application server from the cleared jobs. Affects only completed jobs.
Kill Job	Right-click on a currently running job.	Stops the selected running job after five seconds by killing the server session for the job.
Hide Column	Right-click on a column heading.	Hides the selected column from view in the Job Status Manager window.
Resubmit Job	Right-click on a completed job.	Resubmits the code for a selected job. Available only for completed jobs.
Show	Right-click on any column heading.	Opens a submenu that displays the **`Show Hidden Column`** and **`Show All Columns`** options.
Show All Columns	Right-click on any column heading and select the **`Show`** option.	Displays all of the columns that have been hidden with the **`Hide Column`** option.
Show Hidden Column	Right-click on any column heading and Select the **`Show`** option. Then click on the name of the column that you want to show.	Displays a selected column that has been hidden with the **`Hide Column`** option.

Menu option	Available when you	Description
Sort Ascending	Right-click on any column heading.	Sorts the jobs in ascending order.
Sort Descending	Right-click on any column heading.	Sorts the jobs in descending order.
Toggle Sort	Left-click on any column heading.	Toggles the sort order of the jobs between ascending and descending. Also displays an arrow in the column heading to indicate the sort direction.
View Job	Right-click on any job.	Displays the Process Editor window for the selected job.

Managing Status Handling

Problem

You need to specify a given action based on the status code returned by a job (or, in some cases, a transformation).

Solution

A **`Status Handling`** tab is included in the properties windows for jobs and for some transformations within a job. If the prerequisites for status handling have been met when the job or transformation is executed, the actions that are specified on the **`Status Handling`** tab are carried out. For example, assume that you want to capture statistics about a job, such as the number of records before and after the append of the last table loaded in the job. You will perform the following steps:

1. Display the properties window for the job.
2. Click the **`Status Handling`** tab.
3. Click the **`Errors`** code condition.
4. Click the **`Send Job Status`** action.

Tasks

Use the Status Handling Tab

Perform the following steps to manage status codes and options for a job:

1. On the SAS Data Integration Studio desktop, select the **`Inventory`** tab.
2. Open the Jobs folder in the Inventory tree.
3. If you want to specify status handling for the job, select the job in the project tree and then select **File ▶ Properties** from the menu bar. The properties window for the job is displayed.

   If you want to specify status handling for a transformation with the job, select the job in the project tree and then select **View ▶ View Job** from the menu bar.

The job process flow diagram is displayed in the Process Editor window. In the process flow diagram, select the transformation for which you plan to specify status handling and then select **File ▶ Properties** from the menu bar. The properties window for the transformation is displayed.

4 Click **`Status Handling`** in the properties window for the job or transformation.

5 Click **`New`** to add a row that consists of one code condition and one action.

6 Select the **`Code Condition`** field to display the selection arrow. Click a code condition, such as **`Successful`**.

7 Select the **`Action`** field to display the selection arrow. Click an action that should be performed when the condition is met, such as **`Send Email`**. (The default actions are described in "Maintaining Status Code Conditions and Actions" on page 165.)

8 When you select most actions, the Action Options window is displayed. Use this window to specify options, such as an e-mail address and a message to include in the body of the e-mail. For details about action options, select **`Help`** on the Action Options window.

9 Click **`OK`** when you are finished.

Managing Return Code Check Transformations

Problem

You want to check the status of a transformation that does not contain a **`Status Handling`** tab in a job.

Solution

You can insert status-handling logic where you want it in the process flow diagram for a job by using a Return Code Check transformation. In the example presented here, a Return Code Check transformation will be inserted into the Employee Sort job. Assume the following:

- □ The Employee Sort job is registered in a current metadata repository.
- □ You want to return the status of the Sort transformation in the process flow diagram for the Employee Sort job.
- □ Status messages for the Sort transformation will be written to a file in the directory **`c:\output`**.
- □ Prerequisites for status handling have been met as described in "Prerequisites for Job Status Code Handling" on page 158.

Tasks

Use Return Code Check Transformations

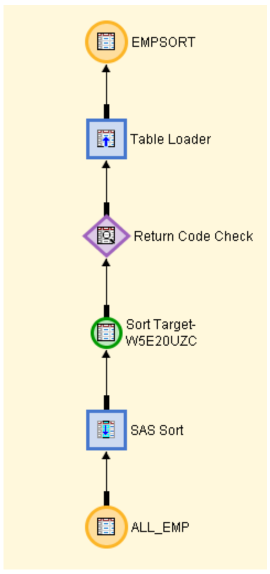
1 On the SAS Data Integration Studio desktop, select the **`Inventory`** tab.

2 In the Inventory tree, open the Jobs folder. Right-click **`Employee Sort`**, and click **`View Job`** in the pop-up menu. The job is opened in the Process Editor window.

3 Open the Process Library tree. Then, open the Control folder.

4 In the Control folder, select, drag, and drop the Return Code Check transformation over the arrow that is between the Sort Target and Loader transformations. The process flow diagram now resembles the following display. The Return Code Check transformation captures the status of the previous transformation in the process flow, which in this case is the Sort transformation.

Display 8.1 Process Flow Diagram with a Return Code Check Transformation

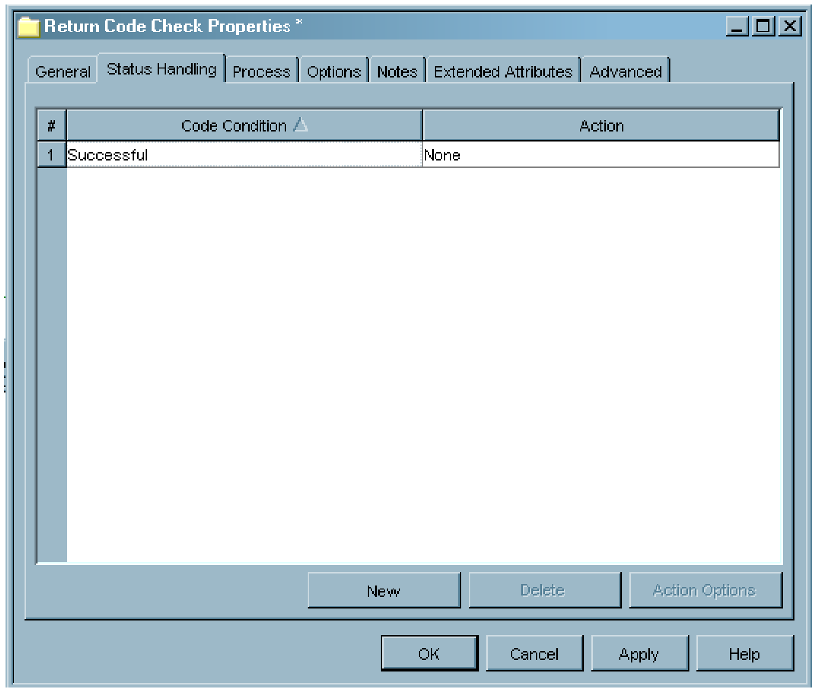


5 Right-click the Return Code Check transformation and select **`Properties`** from the pop-up menu. The properties window is displayed.

6 Click the **`Status Handling`** tab.

7 Click **`New`**. A new Code condition and action row is added, as shown in the following display. The code condition **`Successful`** and the action **`None`** are the default values. If these values were saved without change, no action would be taken whenever the previous transformation in the process flow was successfully executed. For this example, the default action is changed so that a status message is written to a file.

Display 8.2 Status Handling Tab

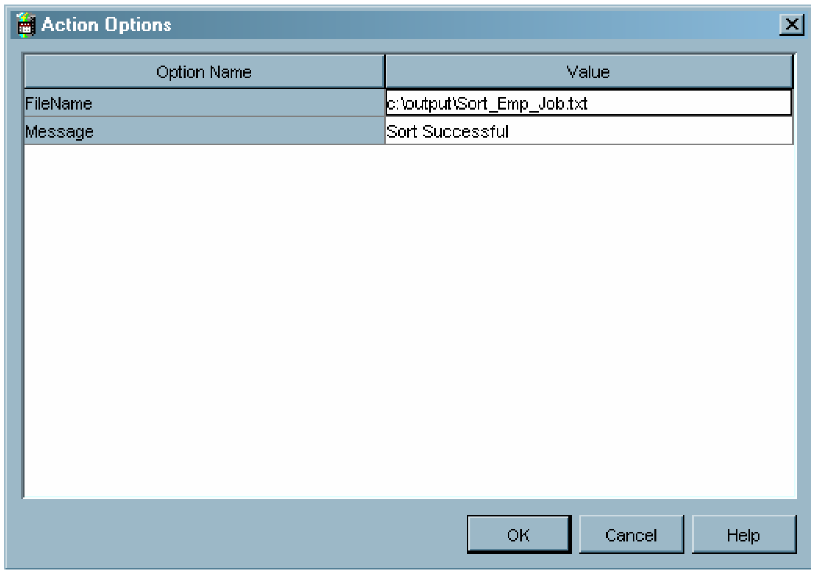


#	Code Condition	Action
1	Successful	None

8 Select the **`Action`** field. A selection arrow appears.

9 Use the selection arrow to select an appropriate action. In this example, the **`Send Entry to Text File`** action is used. An Action Options window displays for this action because you must enter both a fully qualified path to the file that contains the status messages and the text of the message. The options window is shown in the following display.

Display 8.3 Action Options Window



Option Name	Value
FileName	c:\output\Sort_Emp_Job.txt
Message	Sort Successful

10 When you finish entering the options for this action, click **`OK`**. The options are saved, and you are returned to the **`Status Handling`** tab.

11 Click **`OK`** to close the properties window.

12 Right-click the job in the project tree, and select **`Submit Job`**. The server executes the SAS code for the job. If the job completes without error, go to the next section. If error messages appear, respond to the messages and try again.

13 View the status handling output. To verify that a status message was written to a file, go to the directory where the message file was to have been written. In the current example, look for a file with the physical path **`c:\output\Sort_Emp_Job_Status.txt`**. Open the file and view any status messages. In the current example, the messages resembles the following: **`10FEB04:17:40:23 Sort Successful`**. If you can view the appropriate status message, the job produced the desired result.

14 To verify that a status message was written to a file, go to the directory where the message file was to have been written. In the current example, look for a file with the physical path of **`c:\output\Sort_Emp_Job_Status.txt`**. Open the file and view any status messages. In the current example, you should see a message such as: **`10FEB04:17:40:23 Sort Successful`**.

Maintaining Status Code Conditions and Actions

Problem

You want to edit the default conditions and actions that appear in the **`Status Handling`** tab. However, you cannot edit these conditions and actions in the current release. Moreover, user-defined code conditions and actions are not fully supported.

Fortunately, you can prevent an inappropriate action from being displayed in the **`Status Handling`** tab. For example, assume that your site had no plans to implement an event broker. You can prevent all **`Send Event`** actions from showing up in the **`Status Handling`** tab so that users cannot select them.

Solution

To prevent an action from showing up in the **`Status Handling`** tab, use the Configure Status Handling window to delete all code condition and action combinations that include the action that you want to exclude. For example, if you want to exclude all **`Send Event`** actions, you can delete all code condition and action combinations that include **`Send Event`**. You can use this method only when the status handling prerequisites have been met.

Tasks

Manage Default Actions

Perform the following steps to prevent an inappropriate action from being displayed in the **`Status Handling`** tab.

1 From the SAS Data Integration Studio desktop, select **Tools ▶ Configure Status Handling**. The Configure Status Handling window is displayed in update mode.

2 To delete a condition and action combination, in the Available code conditions/action combinations pane, select the desired combination and then select **`Delete`**. The condition and action combination is no longer be available in the **`Status Handling`** tab.

3 Click **`OK`**.

Users can select from a list of return code conditions and actions in the **`Status Handling`** tab for jobs and transformations. The following table provides a list of all the available actions. Note, however, that not all of the actions are available to all transformations. For a list of which actions are available for each type of transformation, see the SAS Data Integration Studio Help.

Table 8.4 Default Actions

Action	Description
Abort	Terminates the job or transformation.
Abort All Processes (Executing and Remaining)	Terminates all of the currently executing and remaining processes.
Abort Remaining Iterations	Terminates all of the remaining iterations of a loop without affecting currently executing iterations.
Abort After Looping	Completes all of the processes in the loop and then terminates the job.
Custom	Enables you to call a macro in a SAS Autocall library to provide user-defined status handling for a job or transformation. The SAS Application Server that executes the job in which custom action code is used must be able to access the relevant Autocall library. If the library contains a macro that is named SENDCUSTOM (a macro that would perform a status handling action), you can use a **`Custom`** action to call that macro. Enter a call to a status handling macro in a SAS Autocall library, such as the following: **`%sendcustom;`**.
Email Exception Report	E-mails an exception report that lists the column name, type of exception, action taken, and related information. Enter one or more recipient e-mail addresses in the Options window for the **`Email Exception Report`** action. Separate multiple e-mail addresses with a semicolon.
Save Exception Report	Saves an exception report to a file. The report lists the column name, type of exception, action taken, and related information. Enter a fully qualified path to the file that contains the report in the Options window for the **`Save Exception Report`** action. The path to the report file must be accessible to the SAS Application Server that is executing the job.
Save Table	Saves status messages to a table. Consecutive messages are appended to the table with a timestamp. Enter the table name in the libref.dataset SAS format, such as **`targetlib.allemp`**. Note that the libref must be assigned before the job or transformation executes.

Action	Description
Send Email	E-mails a message that you specify. Enter one or more recipient e-mail addresses and a message in the options window for the **`Send Email`** action. To specify more than one e-mail address, enclose the group of addresses in parentheses, enclose each address in quotation marks, and separate the addresses with a space, as in **`user1@domain.com`** and **`user2@domain.com`**. Any text in the **`Message`** field that includes white space must be enclosed by single quotation marks so that the mail is processed correctly.
Send Entry to Text File	Saves status messages to a file. Consecutive messages are appended to the file with a timestamp. Enter a fully qualified path to the file that contains the status messages and the text of the message in the Options window for the **`Send Entry to Text File`** action. The path must be accessible to the SAS Application Server that is executing the job.
Send Entry to Dataset	Saves status messages to a SAS data set. Consecutive messages are appended to the data set with a timestamp. Enter a libref for the SAS library where the status messages are to be written, a data set name for the message, and the text of the message in the options window for the **`Send Entry to Dataset`** action. The libref in the **`Libref`** field must be assigned before the job or transformation executes.
Send Event	Sends a status message to an event broker, which sends the message to applications that have subscribed to the broker. The subscribing applications can then respond to the status of the SAS Data Integration Studio job or transformation. For details about the options for the **`Send Event`** action, see the SAS Data Integration Studio Help for the Event Options window.
Send Job Status	Saves status messages to a SAS data set. Consecutive messages are appended to the data set with a timestamp. If the data set does not exist, it is created the first time that the job is executed. Enter a libref for the SAS library where the status messages are to be written and a data set name for the message in the Options window for the **`Send Job Status`** action. The libref in the **`Libref`** field must be assigned before the job or transformation executes. When the **`Send Job Status`** action is selected, the following values are captured: □ job name □ job status □ job return code □ number of records before append of the last table loaded in the job □ number of records after append of the last table loaded in the job □ the library and table name of the last table loaded in the job □ the user who ran the job □ the time the job started and finished

CHAPTER

9

Deploying Jobs

About Deploying Jobs

In a production environment, SAS Data Integration Studio jobs must often be executed outside of SAS Data Integration Studio. For example, a job might have to be scheduled to run at a specified time, or a job might have to be made available as a stored process.

Accordingly, SAS Data Integration Studio enables you to do the following tasks:

- Deploy a job for scheduling; see “About Job Scheduling” on page 170
- Deploy a job as a SAS stored process; see “About SAS Stored Processes” on page 177
- Deploy a job as a SAS stored process that can be accessed by a Web service client; see “About Deploying Jobs for Execution by a Web Service Client” on page 182

You can also deploy a job in order to accomplish the following tasks:

- Divide a complex process flow into a set of smaller flows that are joined together and can be executed in a particular sequence; see “Using Scheduling to Handle Complex Process Flows” on page 174
- Execute a job on a remote host; see “Deploying Jobs for Execution on a Remote Host” on page 175

About Job Scheduling

You can select a job in the Inventory tree or the Custom tree on the SAS Data Integration Studio desktop and deploy it for scheduling. Code is generated for the job and the generated code is saved to a file in a source repository. Metadata about the deployed job is saved to the current metadata repository. After a job has been deployed for scheduling, the person responsible for scheduling jobs can use the appropriate software to schedule the job for execution by a SAS Workspace Server.

The main tasks that are associated with deploying a job for scheduling are as follows:

- □ "Deploying Jobs for Scheduling" on page 171
- □ "Redeploying Jobs for Scheduling" on page 173
- □ "Using Scheduling to Handle Complex Process Flows" on page 174

Prerequisites for Scheduling

To deploy a job for scheduling in SAS Data Integration Studio, you will need the following components:

- □ scheduling server
- □ SAS Workspace Server that is used to deploy jobs for scheduling
- □ source repository for the code that is generated from a SAS Data Integration Studio job

Typically, an administrator installs, configures, and registers the appropriate servers and source code repositories. The administrator then tells SAS Data Integration Studio users which server and source repository to select when deploying jobs for scheduling. For more information, see the scheduling chapters in the *SAS Intelligence Platform: System Administration Guide*.

Deploying Jobs for Scheduling

Problem

You want to schedule a SAS Data Integration Studio job to run in batch mode at a specified date and time.

Solution

Scheduling a job is a two-stage process:

- □ Deploy the job for scheduling. See "Deploy a Job for Scheduling" on page 171.
- □ Schedule the job for execution. See "Schedule a Job" on page 173.

If possible, test a job and verify its output before deploying the job for scheduling.

Tasks

Deploy a Job for Scheduling

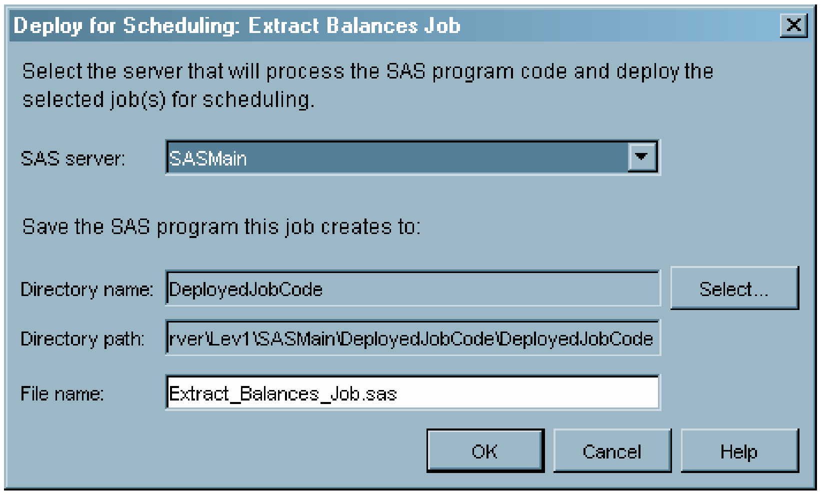
Perform the following steps to deploy a job for scheduling:

1 From the SAS Data Integration Studio desktop, select the **`Inventory`** tab in the tree view.

2 In the Inventory tree, expand the **`Jobs`** folder.

3 Select the job that you want to deploy. Then, right-click to display the pop-up menu, and select **`Deploy for Scheduling`**. The Deploy for Scheduling window displays. If you select only one job, the window resembles the following display.

Display 9.1 Deploy for Scheduling Window for a Single Job



Deploy for Scheduling: Extract Balances Job

Select the server that will process the SAS program code and deploy the selected job(s) for scheduling.

SAS server: SASMain

Save the SAS program this job creates to:

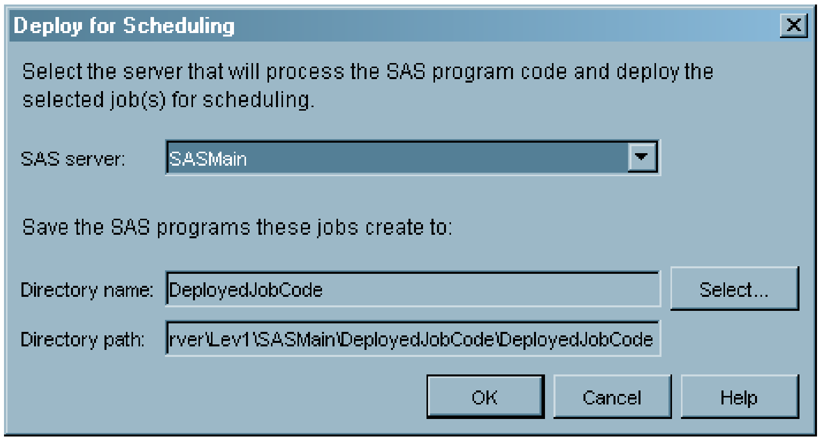
Directory name: DeployedJobCode

Directory path: rver\Lev1\SASMain\DeployedJobCode\DeployedJobCode

File name: Extract_Balances_Job.sas

By default, the deployed job file (in this case, Extract Balances Job.sas) is named after the selected job. If you select more than one job, the window resembles this display.

Display 9.2 Deploy for Scheduling Window for Multiple Jobs



Deploy for Scheduling

Select the server that will process the SAS program code and deploy the selected job(s) for scheduling.

SAS server: SASMain

Save the SAS programs these jobs create to:

Directory name: DeployedJobCode

Directory path: rver\Lev1\SASMain\DeployedJobCode\DeployedJobCode

When you deploy more than one job, a separate file is created for each job that you select. Each deployed job file is named after the corresponding job.

4 In the **`SAS server`** field, accept the default server or select the server that is used to generate and store code for the selected job. The next step is to select the job deployment directory. One or more job deployment directories (source repositories) were defined for the selected server when the metadata for that server was created.

5 Click **`Select`**. The Deployment Directories window is displayed.

6 In the Deployment Directories window, select a directory where the generated code for the selected job will be stored. Then, click **`OK`**. The Deploy for Scheduling window is displayed. The directory that you selected is specified in the **`Directory name`** field.

7 If you selected one job, in the **`File name`** field, you can edit the default name of the file that contains the generated code for the selected job. The name must be unique in the context of the directory specified in the **`Directory name`** field.

8 When you are ready to deploy the job or jobs, click **`OK`**.

Code is generated for the selected job or jobs and is saved to the directory that is specified in the **`Directory name`** field. Metadata about the deployed jobs is saved to the current metadata repository. A status window is displayed and indicates whether the deployment was successful. In the Inventory tree, the icon beside each deployed job includes the image of a small clock. The clock indicates that the jobs are now available for scheduling.

Schedule a Job

After a job has been deployed for scheduling, the person responsible for scheduling can use the appropriate software to schedule the job. For more information, see the scheduling chapters in the *SAS Intelligence Platform: System Administration Guide*.

Redeploying Jobs for Scheduling

Problem

After deploying a job for scheduling and then updating the job, you need to redeploy the job so that the latest version of the job is scheduled. You also need to redeploy jobs if the computing environment changes, for example, when a job is exported from a test environment and is imported into a production environment.

Solution

Use the Redeploy Jobs for Scheduling feature to find any jobs that have been deployed for scheduling, regenerate the code for these jobs, and save the new code to a job deployment directory. The redeployed jobs can then be rescheduled.

Rescheduling a job is a two-stage process:

- Redeploy the job for scheduling. See “Redeploy a Job for Scheduling” on page 174.
- Reschedule the job for execution. See “Reschedule the Job” on page 174.

Tasks

Redeploy a Job for Scheduling

Perform the following steps to redeploy a job for scheduling:

1 From the SAS Data Integration Studio desktop, select **Tools ▶ Redeploy for Scheduling**. Any jobs that have been deployed are found. A dialog box is displayed, citing the number of deployed jobs found and prompting you to click **OK** to continue.

2 Click **OK**. A window is displayed that has various options for maintaining a metadata profile. Code is generated for all deployed jobs and saved to the job deployment directory for the SAS Application Server that is used to deploy jobs.

The regenerated code contains references to servers and libraries that are appropriate for the current metadata repositories. The regenerated jobs are now available for scheduling.

Reschedule the Job

After a job has been redeployed from SAS Data Integration Studio, the person responsible for scheduling can use the appropriate software to reschedule the job. For example, you can use the Schedule Manager in SAS Management Console to reschedule all flows, resubmitting all job flows that have been submitted to the scheduling server. For details, see the online Help for the Schedule Manager in SAS Management Console.

Using Scheduling to Handle Complex Process Flows

Problem

You have a complex job involving joins and transformations from many different tables. You want to reduce the complexity by creating a set of smaller jobs that are joined together and can then be executed in a particular sequence.

Solution

Group all of the jobs in the flow together in a single folder in the Custom tree. Perform the steps in “Schedule Complex Process Flows” on page 174 to deploy and schedule the jobs in the proper sequence.

Tasks

Schedule Complex Process Flows

Perform the following steps to schedule complex process flows:

1 Divide the complex job into a series of smaller jobs that create permanent tables. Those tables can then be used as input for succeeding jobs.

2 Keep all of your jobs in the flow together in a single folder in the Custom tree, and give the jobs a prefix that causes them to be displayed in the appropriate execution order.

3 Deploy the jobs for scheduling.

4 The person responsible for scheduling can use the appropriate software to schedule the jobs to be executed in the proper sequence.

Deploying Jobs for Execution on a Remote Host

Problem

You want to execute one or more SAS Data Integration Studio jobs that process a large amount of data on a remote machine and then save the results to that remote machine. In this case, it might be efficient to move the job itself to the remote machine.

Solution

Administrators must configure the servers that are described in the "Processing Jobs Remotely" section of the *SAS Intelligence Platform: Desktop Application Administration Guide*. A SAS Data Integration Studio user can then use the Deploy for Scheduling window to deploy a job for execution on the remote host. Code is generated for the job and the generated code is saved to a file on the remote host. After a job has been deployed to the remote host, it can be executed by any convenient means.

For example, assume that the default SAS Application Server for SAS Data Integration Studio is called SASMain, but you want a job to execute on another SAS Application Server that is called DEZ_App Server. You will select DEZ_App Server in the Deploy for Scheduling window, and the code that is generated for the job will be local to DEZ_App Server.

Tasks

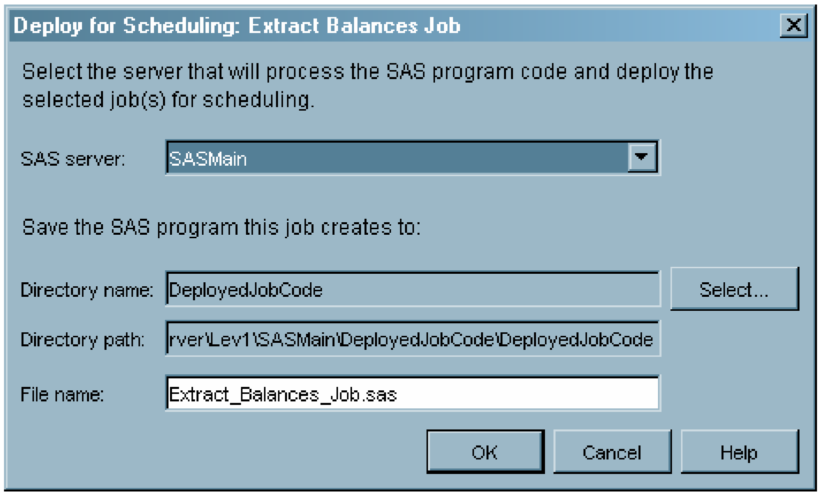
Deploy One or More Jobs for Execution on a Remote Host

Perform the following tasks to deploy jobs for execution on a remote host:

1 From the SAS Data Integration Studio desktop, select **Repositories ▶ Foundation ▶ Jobs** in the Inventory tree.

2 Right-click the job or jobs that you want to deploy. Then, select **`Deploy for Scheduling`**. The Deploy for Scheduling window is displayed. If you selected only one job, the window resembles the following display:

Display 9.3 Scheduling Window for One Job



Deploy for Scheduling: Extract Balances Job

Select the server that will process the SAS program code and deploy the selected job(s) for scheduling.

SAS server: SASMain

Save the SAS program this job creates to:

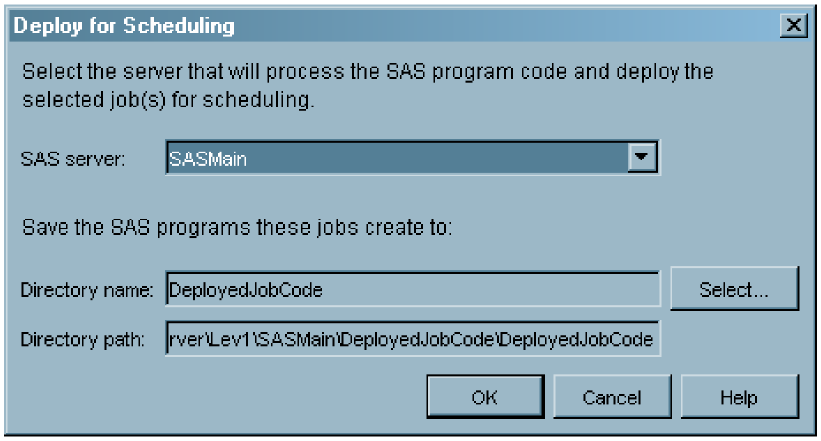
Directory name: DeployedJobCode

Directory path: rver\Lev1\SASMain\DeployedJobCode\DeployedJobCode

File name: Extract_Balances_Job.sas

If you selected more than one job, the window resembles the following display:

Display 9.4 Scheduling Window for More Than One Job



Deploy for Scheduling

Select the server that will process the SAS program code and deploy the selected job(s) for scheduling.

SAS server: SASMain

Save the SAS programs these jobs create to:

Directory name: DeployedJobCode

Directory path: rver\Lev1\SASMain\DeployedJobCode\DeployedJobCode

3 In the drop-down list in the **`SAS Server`** field, select the SAS Application Server that contains the servers on the remote host.

4 In the **`Directory`** field, use the down arrow to select a predefined directory where the generated code for the selected job will be stored.

5 If you selected one job, you can edit the default name of the file that contains the generated code for the selected job in the **`File name`** field. The name must be unique in the context of the directory that is specified above. Click **`OK`** to deploy the job.

If you selected more than one job, SAS Data Integration Studio automatically generates filenames that match the job names. If the files already exist, a message asking whether you want to overwrite the existing files is displayed. Click **`Yes`** to overwrite them. Otherwise, click **`No`**.

Code is generated for the current jobs and saved to the directory that is specified in the **`Directory`** field. Metadata about the deployed jobs is saved to the current metadata repository. In the Inventory tree, the icon beside each deployed job includes the image of a small clock. The clock indicates that the jobs are now available for scheduling. The deployed job can either be scheduled or executed by any convenient means.

About SAS Stored Processes

A SAS stored process is a SAS program that is stored on a server and can be executed as required by requesting applications. You can use stored processes for Web reporting, analytics, building Web applications, delivering result packages to clients or the middle tier, and publishing results to channels or repositories. Stored processes can also access any SAS data source or external file and create new data sets, files, or other data targets supported by the SAS System.

After a job has been deployed as a stored process, any application that can execute a SAS stored process can execute the job. For example, the following applications can execute stored processes for SAS Data Integration Studio jobs:

- SAS Add-In for Microsoft Office: a Component Object Model (COM) add-in that extends Microsoft Office by enabling you to dynamically execute stored processes and embed the results in Microsoft Word documents and Microsoft Excel spreadsheets.
- SAS Enterprise Guide: an integrated solution for authoring, editing, and testing stored processes.
- SAS Information Map Studio: an application can be used to implement information map data sources. Stored processes can use the full power of SAS procedures and the DATA step to generate or update the data in an information map.
- SAS Information Delivery Portal: a portal that provides integrated Web access to SAS reports, stored processes, information maps, and channels.
- Stored Process Service: a Java application programming interface (API) that enables you to execute stored processes from a Java program.
- SAS Stored Process Web applications: Java Web applications that can execute stored processes and return results to a Web browser.
- SAS BI Web Services: a Web service interface to SAS stored processes.

The main tasks that are associated with deploying a Stored Process are as follows:

- "Deploying Jobs as SAS Stored Processes" on page 178
- "Redeploying Jobs to Stored Processes" on page 180
- "Viewing or Updating Stored Process Metadata" on page 181

Prerequisites for SAS Stored Processes

To deploy a SAS Data Integration Studio job as a SAS stored process, you will need the following components:

- A server that can execute SAS stored processes. Stored processes that can be executed by Web service clients require a connection to a SAS Stored Process Server. Other stored processes can be executed on a SAS Stored Process Server or a SAS Workspace Server. For details about how these servers are installed, configured, and registered in a SAS metadata repository, see the *SAS Intelligence Platform: Application Server Administration Guide*.
- A source repository for the stored process that is generated from a SAS Data Integration Studio job.

Typically, an administrator installs, configures, and registers the appropriate servers and source code repositories. The administrator then tells SAS Data Integration Studio users which server and source repository to select when deploying jobs as stored processes.

To use the stored process feature to your best advantage, you should be familiar with stored process parameters, input streams, and result types. For a detailed discussion of stored processes, see “SAS Stored Processes” and “SAS BI Web Services” in the *SAS Integration Technologies: Developer’s Guide*.

Deploying Jobs as SAS Stored Processes

Problem

You want to deploy a job as a SAS stored process so that any application that can execute a SAS stored process can execute the job.

Solution

Deploying and executing a job as a stored process is two-stage procedure:

- ☐ Deploy the job as a stored process. See “Deploy a Job as a Stored Process” on page 178.
- ☐ Execute the stored process. See “Execute the Stored Process for a Job” on page 180.

If possible, test a job and verify its output before deploying the job as a stored process.

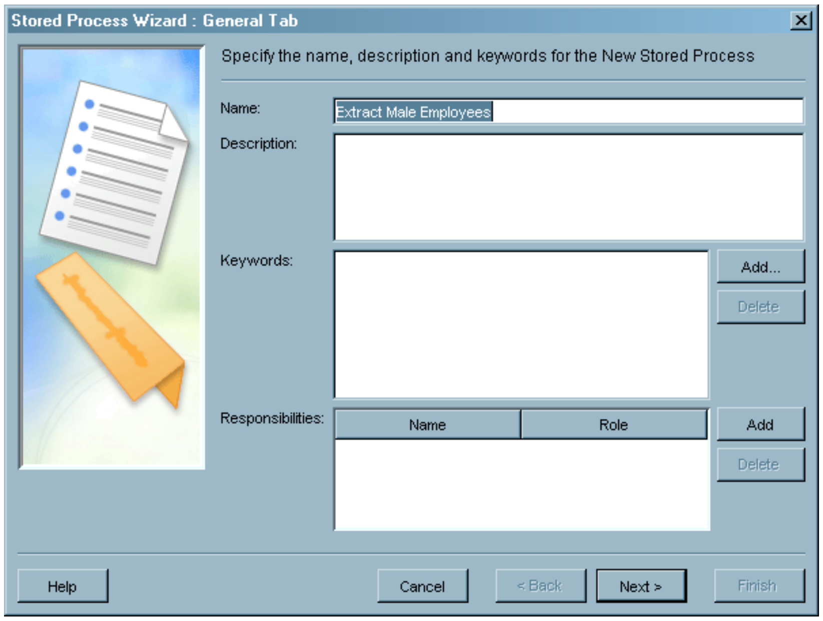
Tasks

Deploy a Job as a Stored Process

Perform the following steps to deploy a job as a stored process:

1 In the Inventory tree or the Custom tree on the SAS Data Integration Studio desktop, right-click the job for which you will like to generate a stored process. Then select **Stored Process ▶ New** from the pop-up menu. The first window of the Stored Process wizard is displayed.

Display 9.5 First Window of the Stored Process Wizard


Stored Process Wizard : General Tab

Specify the name, description and keywords for the New Stored Process

Name: Extract Male Employees

Description:

Keywords:

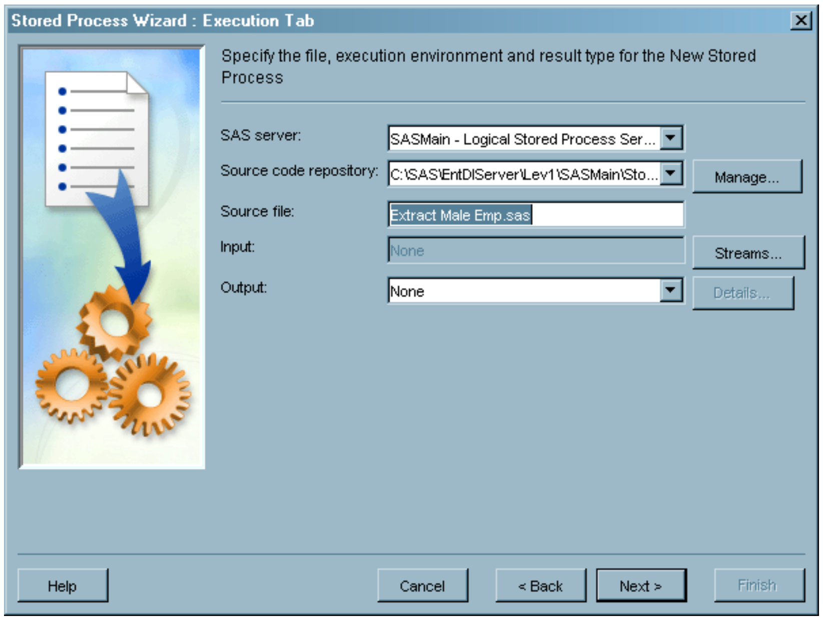
Responsibilities:

Name	Role

2 In the first window, enter a descriptive name for the stored process metadata. You might want to use a variation of the job name. Enter other information as desired. For details about the fields in this window, select **`Help`**. When finished, select **`Next`**. The second window of the wizard is displayed.

3 In the second window, specify a SAS server, a source repository, a source filename, any input stream, and any output type (result type) for the new stored process. The following display shows some sample values for this window.

Display 9.6 Second Window of the Stored Process Wizard


Stored Process Wizard : Execution Tab

Specify the file, execution environment and result type for the New Stored Process

SAS server: SASMain - Logical Stored Process Ser...

Source code repository: C:\SAS\EntDIServer\Lev1\SASMain\Sto...

Source file: Extract Male Emp.sas

Input: None

Output: None

4 The main fields in this window are as follows:
   - □ **SAS server:** a SAS Workspace Server or a SAS Stored Process Server.
   - □ **Source code repository:** a location, such as a directory, that contains stored processes. The physical path that is specified in this field is associated with the server that is specified in the SAS server field. You must have write access to this repository.
   - □ **Source file:** name of the stored process file that is to be generated from the job that was selected.
   - □ **Input:** specifies the type of input for the stored process. This is automatically set to None if you are using a SAS Workspace Server or if you are using a SAS Stored Process Server with no defined streams.
   - □ **Output:** specifies the type of results produced by the stored process. A list of result types is provided for you. The list of result types changes depending on whether you select a SAS Workspace Server or a SAS Stored Process Server.

   For details about the fields in this window, select **Help**. When finished, select **Next**.

5 The third window of the wizard is displayed. In the third window, specify any parameters for the stored process. For details about the fields in this window, select **Help**. When finished, select **Finish**.

   A stored process is generated for the current job and is saved to the source repository. Metadata about the stored process is saved to the current metadata repository. In the Inventory tree, the icon beside the selected job includes the image of a blue triangle. The blue triangle indicates that a stored process has been generated for the job. A metadata object for the stored process is added to the **Stored Process** folder in the Inventory tree.

Execute the Stored Process for a Job

After a job has been deployed as a stored process, the person responsible for executing that job can use the appropriate application to access and run the job.

Redeploying Jobs to Stored Processes

Problem

After generating a stored process for a job and then updating the job, you want to redeploy the stored process so that it matches the latest version of the job. You also need to redeploy stored processes if the computing environment changes, as when a stored process is exported from a test environment and is imported into a production environment, for example.

Solution

You can select a job for which a stored process has been generated, regenerate code for the job, and update any stored processes associated with the selected job. See "Redeploy a Selected Job with a Stored Process" on page 181.

Alternatively, you can use the Redeploy Jobs to Stored Processes feature to regenerate the code for most jobs with stored processes and update any stored processes associated with these jobs. Each redeployed stored process then matches the current version of the corresponding job. See "Redeploy Most Jobs with Stored Processes" on page 181.

Tasks

Redeploy a Selected Job with a Stored Process

Perform the following steps to select a job for which a stored process has been generated, regenerate code for the job, and update any stored processes associated with the selected job:

1. Open the `Jobs` folder in the Inventory tree.
2. Right-click the job metadata for a stored process.
3. Select **Stored Process** ▶ ***job_name*** ▶ **Redeploy** from the pop-up menu.

Redeploy Most Jobs with Stored Processes

Perform the following steps to regenerate the code for most jobs with stored processes and update any stored processes associated with these jobs.

Note: The Redeploy Jobs to Stored Processes feature does not redeploy a job that has been deployed for execution by a Web service client. △

1. From the SAS Data Integration Studio desktop, select **Tools** ▶ **Redeploy Jobs to Stored Processes**.
2. A dialog box is displayed, citing the number of stored processes found and prompting you to click `OK` to continue.

For each job that has one or more associated stored processes, the code is regenerated for that job. For each stored process associated with a job, the generated code is written to the file associated with the stored process. The regenerated code contains references to servers and libraries that are appropriate for the current metadata repositories.

Viewing or Updating Stored Process Metadata

Problem

You want to update or delete the metadata for a stored process.

Solution

Locate the metadata for the stored process in the `Stored Process` folder of the Inventory tree. Display the properties window and update the metadata.

The SAS server and source repository for a stored process cannot be changed. If you need to change these values, create a new stored process and specify a different server and source repository.

Tasks

Update the Metadata for a Stored Process

Perform the following steps to update the metadata for a stored process that was generated for a SAS Data Integration Studio job:

1. In the Inventory tree on the SAS Data Integration Studio desktop, locate the **`Stored Process`** folder.
2. Locate the metadata for the stored process that you want to update.
3. To delete the metadata for a stored process, right-click the appropriate process and select **`Delete`**. (The physical file that contains the stored process code is not deleted; only the metadata that references the file is deleted.)

   To view or update the metadata for a stored process, right-click the appropriate process and select **`Properties`**. A properties window for the stored process is displayed.
4. View or update the metadata as desired. For details about the tabs in this window, select **`Help`**.

About Deploying Jobs for Execution by a Web Service Client

A Web service is an interface that enables communication between distributed applications, even if the applications are written in different programming languages or are running on different operating systems. In SAS Data Integration Studio, a job can be selected in the Inventory tree or the Custom tree and deployed as a SAS stored process that can be executed by a Web service client.

You might want to deploy a job for execution by a Web service client when the job meets all of the following criteria:

- The job must be accessed across platforms.
- The amount of data to be input and output is not large.
- Any input from the Web service client, or output to the Web service client, can be formatted as an XML table.

The main tasks that are associated with deploying a job for execution by a Web service client are as follows:

- Review the requirements for a job that will be deployed for execution by a Web service client. See “Requirements for Jobs That Can Be Executed by a Web Service Client” on page 183.
- Create the job. See “Creating a Job That Can Be Executed by a Web Service Client” on page 185.
- Deploy the job. See “Deploying Jobs for Execution by a Web Service Client” on page 188.

After the job has been deployed, the person responsible for executing the deployed job can use the appropriate Web service client to access and execute the job. See “Using a Web Service Client to Execute a Job” on page 191.

Requirements for Jobs That Can Be Executed by a Web Service Client

Overview of Requirements

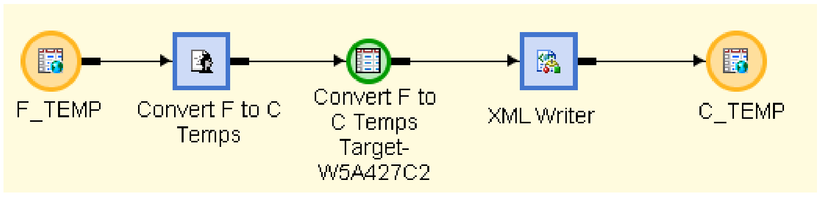
In addition to the “Prerequisites for SAS Stored Processes” on page 177, a SAS Data Integration Studio job that can be executed by a Web service client must meet the requirements that are described in this section.

Process Flow Requirements

If a job does not need to receive input from the client and does not have to send output to the client, then the job does not have to meet any additional requirements before it is deployed for execution by a Web service client. It can be deployed as described in “Deploying Jobs for Execution by a Web Service Client” on page 188.

Usually, however, a job receives input from the client and sends output to the client. For example, suppose that you wanted to deploy a job that will enable a user to provide a temperature value in Fahrenheit and obtain the corresponding value in Celsius. The following display shows how you can specify such a process flow in SAS Data Integration Studio.

Display 9.7 Process Flow for a Temperature Conversion Job



A job that obtains input from and sends output to a Web service client must meet the following requirements before it is deployed:

- The job can receive zero or more inputs from the Web service client that executes the job. The preceding temperature conversion job receives one input from the client, as specified in F_TEMP.
- The job can send zero or one output to the client that executes the job. The job shown here sends one output to the client, as specified in C_TEMP.
- Input to the job from the client, and output from the job to the client, must be in XML table format. In this job, F_TEMP and C_TEMP are XML tables.
- XML tables that specify client input or output in the job must be members of a SAS XML library.
- The XML table for a client input can have an XMLMap associated with it through the library. An XMLMap can help the XML LIBNAME engine to read the table. However, the XML table that specifies a client output cannot have an XMLMap associated with it through the library.
- The XML table for each client input or output in the job must have a unique libref.
- The client output in the process flow must have a libref named _WEBOUT.

- The XML table for each client input or output in the job must be configured as a Web stream. F_TEMP and C_TEMP are configured as Web streams, as indicated by the blue circles on their icons. (See "Web Streams for Web Client Inputs and Outputs" on page 185.)

Data Format for Web Client Inputs and Outputs

All Web services exchange information in XML format. SAS Data Integration Studio jobs are optimized for structured data such as tables. Accordingly, exchanges between a SAS Data Integration Studio job and a Web service client must be in XML table format.

For example, the following sample code shows an XML table that is appropriate for the client **`input`** in our temperature conversion job.

Example Code 9.1 An XML Table for One Temperature Value in Fahrenheit

```
<?xml version="1.0" encoding="windows-1252" ?>
<FAHRENHEIT>
<F_TEMP>
<DEGREE-F> </DEGREE-F>
</F_TEMP>
</FAHRENHEIT>
```

The XML code in this example can be stored in an XML file named **`f_temps.xml`**. The next example code shows an XML table that is appropriate for the client **`output`** in the temperature conversion job.

Example Code 9.2 An XML Table for One Temperature Value in Celsius

```
<?xml version="1.0" encoding="windows-1252" ?>
<CELSIUS>
<
C_TEMP>
<DEGREE-C> </DEGREE-C>
</C_TEMP>
</CELSIUS>
```

The preceding XML code could be stored in an XML file named **`c_temps.xml`**.

Later these tables could be registered with the XML source designer, and the table metadata can be used to create a process flow in SAS Data Integration Studio.

Libraries and Librefs for Web Client Inputs and Outputs

A job that is deployed for execution by a Web service client has a number of requirements that are related to libraries and librefs.

A SAS XML library is a library that uses the SAS XML LIBNAME engine to access an XML file. In the process flow for the job, the XML tables that specify input from the client or output to the client must be members of a SAS XML Library. For example, suppose that you had created the XML tables that are described in the previous section. You can create one SAS XML Library for the input table (**`f_temps.xml`**) and another SAS XML Library for the output table (**`c_temps.xml`**).

Each library has a libref, an alias for the full physical name of a SAS library. A SAS LIBNAME statement maps the libref to the full physical name. The process flow for a job that is deployed for execution by a Web service client has two requirements in regard to librefs:

- The XML table for each client input and output must have a unique libref. When the job is deployed for execution by a Web service client, the input and output streams will be named after the librefs of the inputs and the output. Different streams in the job cannot have the same librefs.

- □ The client output must have a libref named **`_WEBOUT`**.

For example, if you created a SAS XML Library for the input file **`f_temps.xml`** and another SAS XML Library for the output file **`c_temps.xml`**, you can specify librefs of **`f_temps`** and **`_WEBOUT`**, respectively. In that case, there will be no duplicate librefs for client inputs and outputs, and the libref for the output will be **`_WEBOUT`**, as required.

If needed, you can change the libref in the metadata for a library. In general, to change the libref, right-click the library and select **`Properties`** from the pop-up menu. Click the **`Options`** tab and update the **`Libref`**field.

Web Streams for Web Client Inputs and Outputs

In the process flow for a job that will be deployed for execution by a Web service client, the XML table for each client input or output must be configured as a Web stream. To configure an XML table as a Web stream, right-click the table in the process flow and then select **`Web Stream`** from the pop-up menu. An extended attribute named **`ETLS_WEB_STREAM`** is added to the selected table. A blue circle icon is added to the icon for the table.

(Optional) Parameters for User Input

Suppose that you wanted the Web client user to supply a character or string value to the Web service. In that case, you will:

- □ Add the appropriate column or columns to the Web client input table in the job. See “Data Format for Web Client Inputs and Outputs” on page 184.
- □ When you deploy the job, use the Web Service Wizard to add a parameter to the Web service. See “Deploying Jobs for Execution by a Web Service Client” on page 188.

Creating a Job That Can Be Executed by a Web Service Client

Problem

You want to create a SAS Data Integration Studio job that can be executed by a Web service client.

Solution

Create a SAS Data Integration Studio job that meets the requirements that are described in “Requirements for Jobs That Can Be Executed by a Web Service Client” on page 183.

Tasks

Create a Job That Can Be Executed by a Web Service Client

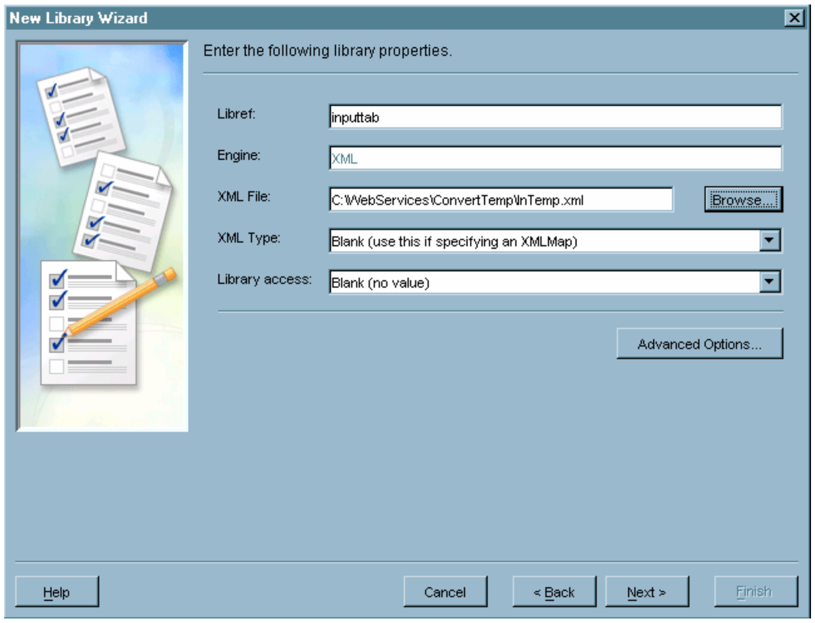
Perform the following steps to create a job that can be executed by a Web service client. The temperature conversion job that is described in “Process Flow Requirements” on page 183 is used as an example.

1 Use an XML editor to create an XML table for each input from the Web service client. Include test values in the input tables, if desired. Save each table to a separate file. For additional information, see “Data Format for Web Client Inputs and Outputs” on page 184.

2 Use an XML editor to create an XML table for the output to the Web service client. Save that table to a file.

3 Using the New Library wizard for SAS XML Libraries, create a separate XML library for each input from the Web service client, or a unique libref for each input, or both. In the following display, the input XML library has a libref of **`inputtab`**.

Display 9.8 New Library Wizard for inputtab



Enter the following library properties.

Libref: inputtab

Engine: XML

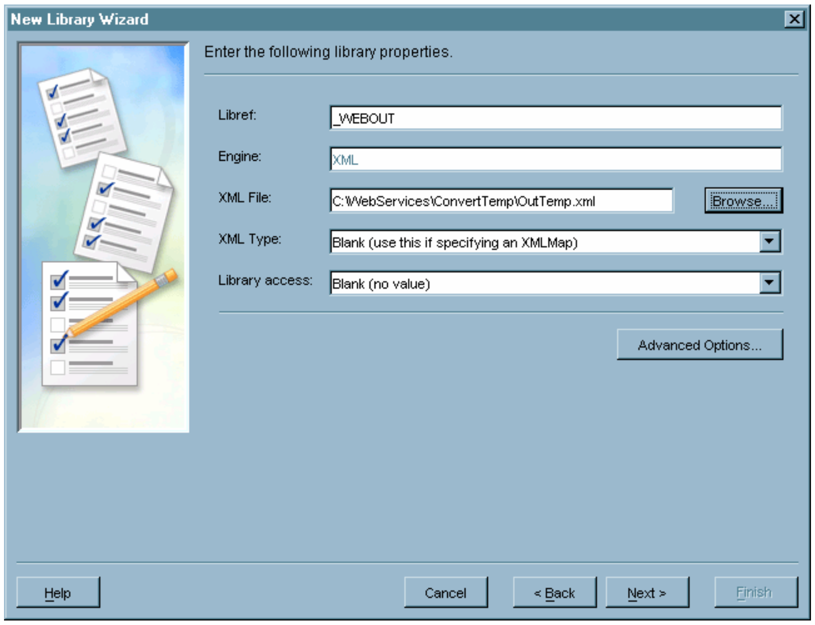
XML File: C:\WebServices\ConvertTemp\InTemp.xml

XML Type: Blank (use this if specifying an XMLMap)

Library access: Blank (no value)

4 Using the New Library wizard for SAS XML Libraries, create an XML library with a libref of **`_WEBOUT`** for the output to the Web service client. The library must have a libref of **`_WEBOUT`**; otherwise, the stream will be ignored.

Display 9.9 New Library Wizard for _WEBOUT



Enter the following library properties.

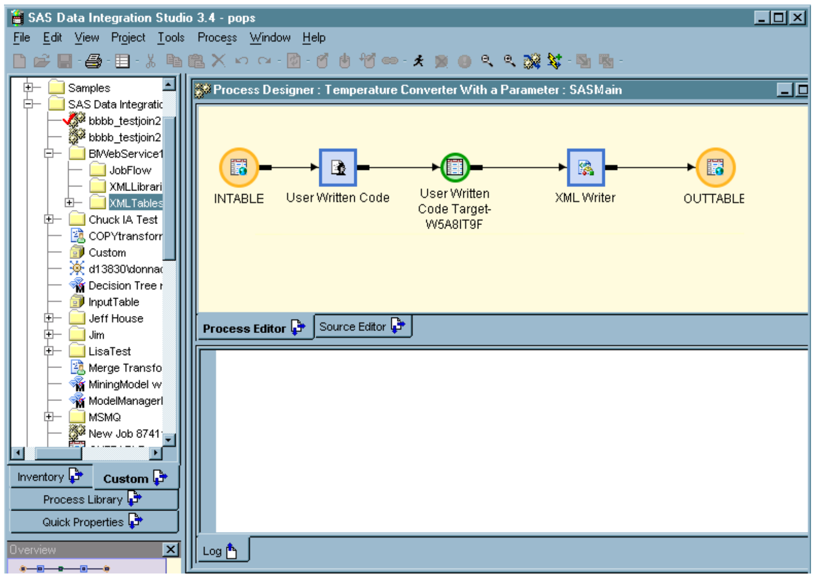
Libref: _WEBOUT
Engine: XML
XML File: C:\WebServices\ConvertTemp\OutTemp.xml
XML Type: Blank (use this if specifying an XMLMap)
Library access: Blank (no value)

5 Use **Tools** ▶ **Source Designer** to register both the **`InputTable`** and **`OutputTable`** libraries. Perform the following steps for both libraries:

- **a** Select **`XML --- All Documents`** from XML Sources. Click **`Next`**.
- **b** Select the library.
- **c** Select **`All Tables`**. Click **`Next`**.
- **d** Select the appropriate folder. Click **`Next`**. Click **`Finish`**.

6 Using the Process Designer window, create a process flow for the Web services job, such as the process flow shown in the temperature conversion job. Use the XML table metadata from the previous step to specify the Web service client input and output.

Display 9.10 Specify Web Service Client Input and Output


SAS Data Integration Studio 3.4 - pops
File Edit View Project Tools Process Window Help
Process Designer : Temperature Converter With a Parameter : SASMain
INTABLE
User Written Code
User Written Code Target-W5A8IT9F
XML Writer
OUTTABLE
Process Editor
Source Editor
Log
Inventory
Custom
Process Library
Quick Properties
Overview

7 If the metadata for each client input table points to an XML table with test values, you can test the job in SAS Data Integration Studio. Run the job, and then use the View Data window to verify that the values in the client output table are correct. If not, troubleshoot and correct the job.

8 Configure the client input and output as Web streams. Right-click a client input in the process flow and then select **Web Stream** from the pop-up menu. Repeat for all inputs and the output in the job.

9 Save and close the job.

Note: After the job is deployed, and the Web client executes the job, any physical table specified in the metadata for a Web stream input or output is ignored, and data submitted by the client is used instead. △

Deploying Jobs for Execution by a Web Service Client

Problem

You want to deploy a SAS Data integration Studio job for execution by a Web service client.

Solution

Select the job and deploy it as described in the following section. It is assumed that the selected job meets the requirements that are described in “Requirements for Jobs That Can Be Executed by a Web Service Client” on page 183.

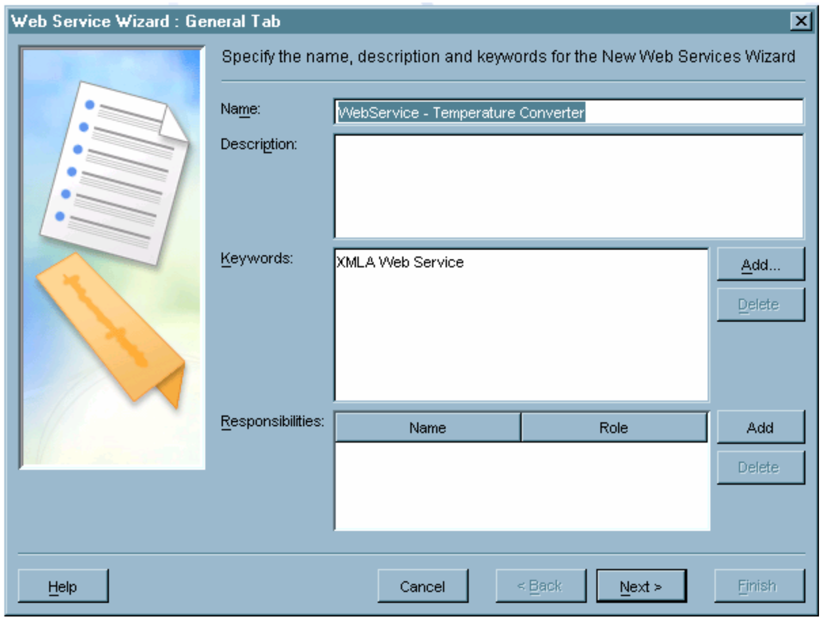
Tasks

Deploy a Job for Execution by a Web Service Client

Perform the following steps to deploy a job for execution by a Web service client. The temperature conversion job that is described in "Process Flow Requirements" on page 183 is used as an example.

1 In the Inventory tree or the Custom tree on the SAS Data Integration Studio desktop, right-click the job that you want to deploy for execution by a Web service client. Then select **Web Service New** from the pop-up menu. The first window of the Web Service wizard displays:

Display 9.11 General Tab


Web Service Wizard : General Tab

Specify the name, description and keywords for the New Web Services Wizard

Name: WebService - Temperature Converter

Description:

Keywords: XMLA Web Service

Responsibilities:

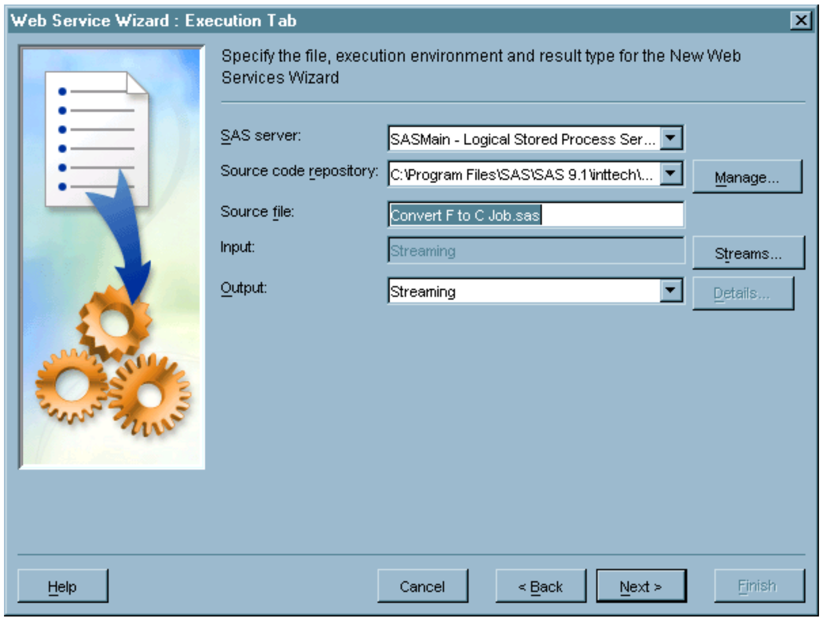
Name	Role

In the first window, the **Name** field specifies the name of the job that you are about to deploy. The name of the selected job (**Convert F to C Job**) is displayed by default. Note the keyword **XMLA Web Service**, which is the required keyword for a job that can be executed by a Web service client.

2 In the first window, enter a descriptive name for the stored process that will be generated from the selected job. You can use a variation of the job name. For example, if the selected job was named **Convert F to C Job**, the metadata for the stored process can be **Convert F to C Service**.

3 In the second window, specify a SAS Stored Process Server, a source repository, a source filename, any input stream, and any output type (result type) for the new stored process. The following display shows some example values for this window.

Display 9.12 Execution Tab


Web Service Wizard : Execution Tab

Specify the file, execution environment and result type for the New Web Services Wizard

SAS server: SASMain - Logical Stored Process Ser...

Source code repository: C:\Program Files\SAS\SAS 9.1\inttech\...

Source file: Convert F to C Job.sas

Input: Streaming

Output: Streaming

4 The main fields in this window are as follows:
  - **SAS server**: for Web services, only SAS Stored Process Servers are available for selection.
  - **Source repository**: a location, such as a directory, that contains stored processes. The physical path that is specified in this field is associated with the server that is specified in the SAS server field. You must have write access to this repository.
  - **Source file**: name of the stored process file that is to be generated from the selected job..
  - **Input**: specifies the type of input for the stored process. This is automatically set to **Streaming** if you are deploying a job with one or more Web Stream inputs.
  - **Output**: specifies the type of results roduced by the stored process. This is automatically set to **Streaming** if you are deploying a job with a Web Stream output.

  For details about the fields in this window, select the **Help** button. When you are finished, select **Next**.

5 In the third window, specify any parameters for the stored process. For details about the fields in this window, select the **Help** button. When finished, select **Finish**.

A stored process that can be executed by a Web service client is generated for the current job and saved to the source repository. Metadata about the stored process is saved to the current metadata repository. In the Inventory tree, the icon beside the selected job includes the image of a blue triangle. The blue triangle indicates that a stored process has been generated for the job. A metadata object for the stored process is added to the **Stored Process** folder in the Inventory tree.

Using a Web Service Client to Execute a Job

Problem

You want to execute a SAS Data Integration Studio job that was deployed for execution by a Web service client.

Solution

Use a Web service client to execute the stored process that was generated from the job.

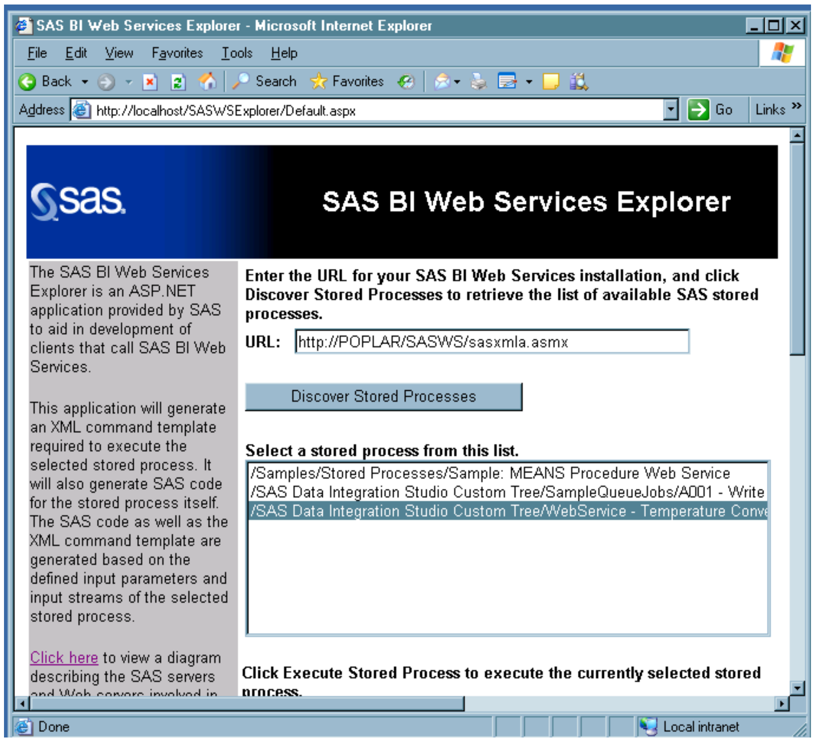
Tasks

Use a Web Service Client to Execute a Job

Perform the following steps to execute the stored process that was generated from a SAS Data Integration Studio job. The SAS BI Web Services Explorer is used as an example of a Web service client. The example job is the temperature conversion job that is described in "Process Flow Requirements" on page 183.

1 Start the Web service client and display a list of stored processes that are available for execution. For example, in the SAS BI Web Services Explorer, you can click **`Discover Stored Processes`**. After the request is submitted, the response looks similar to the following display.

Display 9.13 Stored Processes in SAS BI Web Services Explorer


SAS BI Web Services Explorer - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/SASWSExplorer/Default.aspx Go Links
SAS BI Web Services Explorer
Discover Stored Processes
Done
Local intranet

The SAS BI Web Services Explorer is an ASP.NET application provided by SAS to aid in development of clients that call SAS BI Web Services.

This application will generate an XML command template required to execute the selected stored process. It will also generate SAS code for the stored process itself. The SAS code as well as the XML command template are generated based on the defined input parameters and input streams of the selected stored process.

Click here to view a diagram describing the SAS servers and Web servers involved in

Enter the URL for your SAS BI Web Services installation, and click Discover Stored Processes to retrieve the list of available SAS stored processes.

URL: http://POPLAR/SASWS/sasxmla.asmx

Select a stored process from this list.

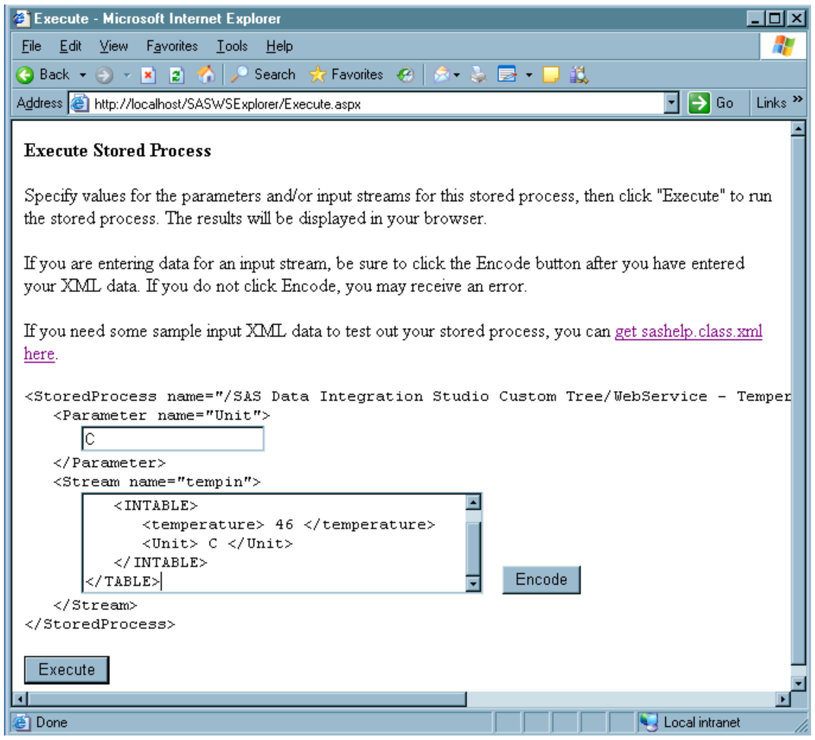
/Samples/Stored Processes/Sample: MEANS Procedure Web Service
/SAS Data Integration Studio Custom Tree/SampleQueueJobs/A001 - Write
/SAS Data Integration Studio Custom Tree/WebService - Temperature Conve

Click Execute Stored Process to execute the currently selected stored process.

2 Select the stored process that you want to execute. If the stored process requires input, the Web service client will display one or more input areas.

3 Enter any required input. The input must be in the format described in “Data Format for Web Client Inputs and Outputs” on page 184. For example, suppose that you selected the stored process for the temperature conversion job. In that case, you will paste an XML table that contains a parameter and a temperature value, as shown in the next display.

Display 9.14 Input Is One Value in an XML Table


Execute - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/SASWSExplorer/Execute.aspx
Go Links
Encode
Execute
Done
Local intranet

Execute Stored Process

Specify values for the parameters and/or input streams for this stored process, then click "Execute" to run the stored process. The results will be displayed in your browser.

If you are entering data for an input stream, be sure to click the Encode button after you have entered your XML data. If you do not click Encode, you may receive an error.

If you need some sample input XML data to test out your stored process, you can get sashelp.class.xml here.

```
<StoredProcess name="/SAS Data Integration Studio Custom Tree/WebService - Temper
   <Parameter name="Unit">
      C
   </Parameter>
   <Stream name="tempin">
         <INTABLE>
            <temperature> 46 </temperature>
            <Unit> C </Unit>
         </INTABLE>
      </TABLE>
   </Stream>
</StoredProcess>
```

4 Submit the input and execute the stored process. For example, in the SAS BI Web Services Explorer, you will click **Encode** to encode the input, and then click **Execute** to execute the stored process.

The stored process executes. If an output value is returned, it is displayed in the Web service client. For example, the stored process for the temperature conversion job converts a Celsius input value and displays a Fahrenheit output value in the client.

CHAPTER 10 Working with Transformations

About Transformations

A transformation is a metadata object that specifies how to extract data, transform data, or load data into data stores. Each transformation that you specify in a process flow diagram generates or retrieves SAS code. You can also specify user-written code in the metadata for any transformation in a process flow diagram. For information about user-written code in transformations, see Chapter 11, "Working with Generated Code," on page 207 and Chapter 12, "Working with User-Written Code," on page 215.

You can find transformations in the Process Library tree on the SAS Data Integration Studio desktop. The transformations that are supplied with SAS Data Integration Studio are divided into the following categories:

- □ Access
- □ Analysis
- □ Archived Transforms
- □ Control

- □ Data Transforms
- □ Output
- □ Publish
- □ SPD Server Dynamic Cluster

See “Process Library Tree” on page 39 for information about the transformations included in these categories. You can create additional transformations and store them in the User Defined section of the Process Library tree.

Each transformation comes with an appropriate set of options. You can set them on the **`Options`** tab of the properties window for the transformation. You can also decide whether to you should send the output of the transformation to a temporary table or to a permanent table. See “Manage Temporary and Permanent Tables for Transformations” on page 239.

Once you have located the transformation that you need, you can drag it from the Process Library tree and drop it into a job. You can view and update the metadata for any transformation that is included in a process flow diagram, as described in “Viewing or Updating the Metadata for Transformations” on page 197. You can also import and export the transformation metadata when the jobs that contain them are imported or exported. The metadata is simply treated as a part of the job.

Viewing the Code for a Transformation

Problem

You want to view the code for a transformation that is included in an existing SAS Data Integration Studio job.

Solution

You can view or update the metadata for a transformation in the transformation’s Source Editor window. This window is available only when the transformation is included in a SAS Data Integration Studio job.

Tasks

Access the Code in a Transformation

You can access the code in a transformation that is included in a SAS Data Integration Studio job. Perform the following steps.

1. Open an existing SAS Data Integration Studio job.
2. Right-click the transformation in the Process Designer window that contains the code that you want to review. Then, click **`View Step Code`**.
3. Review the code that is displayed in the Source Editor window for the selected transformation.

Viewing or Updating the Metadata for Transformations

Problem

You want to view or update the metadata for a transformation. This metadata can include the metadata for column mappings and the metadata that specifies whether you or SAS Data Integration Studio will supply the code for the transformation.

Solution

You can view or update the metadata for a transformation in the transformation's properties window. This window is available only when the transformation is included in a SAS Data Integration Studio job.

Tasks

Access the Metadata for a Transformation

You can access the metadata for a transformation that is included in a SAS Data Integration Studio job. Perform the following steps.

1 Open an existing SAS Data Integration Studio job.
2 Right-click the transformation in the Process Designer window that contains the metadata that you need to review or update. Then, click **`Properties`**.
3 Click the appropriate tab to view or update the desired metadata.

For details about the metadata that is maintained on a particular tab, click **`Help`** on that tab. The Help topics for complex tabs often include task topics that can help you perform the main tasks that are associated with that tab. Updates to transformation metadata are not reflected in the output for that transformation until you rerun the job in which the transformation appears, as described in "Submitting a Job for Immediate Execution" on page 147.

Additional Information

For details about maintaining column mappings, see "Creating and Maintaining Column Mappings" on page 197. For information about maintaining generated transformations, see "Maintaining a Generated Transformation" on page 232.

Creating and Maintaining Column Mappings

Problem

You want to create or maintain the column mappings between the source tables and the target tables in a SAS Data Integration Studio job.

Solution

You create or maintain column mappings in the **Mapping** tab in the properties window for a transformation. You can work with mappings in the following ways:

- □ Create automatic column mappings.
- □ Create one-to-one column mappings.
- □ Create derived column mappings.
- □ Create mappings with multiple columns.
- □ Delete mappings.

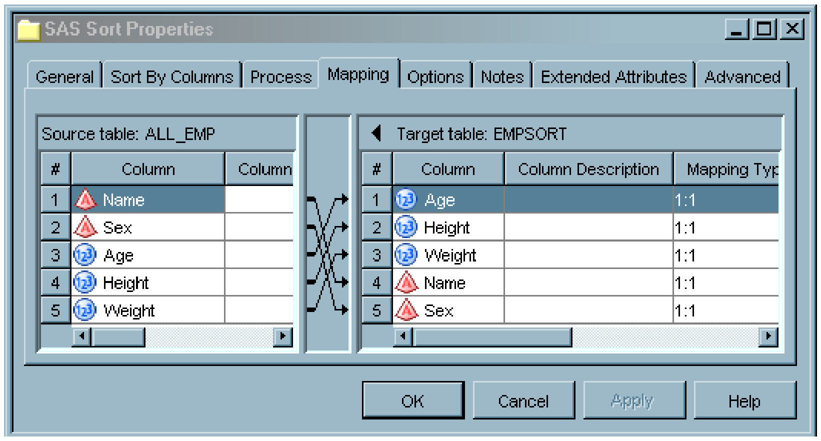
You can also modify the columns in the **Target table** field of the **Mapping** tab. For information, see "Maintaining Column Metadata" on page 98.

Tasks

Create Automatic Column Mappings

You can review the mappings that are automatically generated when a transformation is submitted for execution in the context of a SAS Data Integration Studio job. The mappings are depicted on the **Mapping** tab of the transformation, as shown in the following display.

Display 10.1 Automatic Column Mappings



Source table: ALL_EMP

#	Column	Column
1	Name	
2	Sex	
3	Age	
4	Height	
5	Weight	

Target table: EMPSORT

#	Column	Column Description	Mapping Typ
1	Age		1:1
2	Height		1:1
3	Weight		1:1
4	Name		1:1
5	Sex		1:1

The arrows in the preceding display represent mappings that associate source columns with target columns. By default, SAS Data Integration Studio automatically creates a mapping when a source column and a target column have the same column name, data type, and length. Events that trigger automatic mapping include:

- □ dropping a source and a target in the drop zones for a transformation template
- □ clicking **Propagate** in the toolbar or in the pop-up menu in the Process Designer window
- □ clicking **Quick Map** or **Quick Propagate** in the pop-up menu for the **Mapping** tab of the transformation

SAS Data Integration Studio might not be able to automatically create all column mappings that you need in a transformation.

SAS Data Integration Studio automatically creates a mapping when a source column and a target column have the same column name, data type, and length. However, even though such mappings are valid, they might not be appropriate in the current job.

You can also disable or enable automatic mapping for a transformation. For example, suppose that both the source table and the target table for a transformation have two columns that have the same column name, data type, and length, as shown in the preceding display. These columns are mapped automatically unless you disable automatic mapping for the transformation. If you delete the mappings between these columns, the mappings are restored upon a triggering event, such as selecting the **Propagate**, **Quick Map**, or **Quick Propagate** options.

The only way to prevent automatic mappings is to disable automatic mapping for the transformation. To disable automatic mapping for a selected transformation, right-click the transformation and deselect **Automap** from the pop-up menu. To restore automatic mapping for a selected transformation, right-click the transformation and select **Automap** from the pop-up menu.

Note: If you disable automatic mapping for a transformation, you must maintain its mappings manually. △

Create One-to-One Column Mappings

You need to manually map between a column in the source table and a column in the target table. Perform the following steps to map between two columns:

1 Open the **Mapping** tab in the properties window for the transformation.
2 Select the column in the source table.
3 Select the column in the table.
4 Right-click an empty space in the **Mapping** tab. Then, click **New Mapping** in the pop-up menu. An arrow appears between the two selected columns. This arrow represents the new mapping.
5 Click **OK** to save the new mapping and close the properties window.

You can also create a mapping in the **Mapping** tab by clicking on a source column and dragging a line to the appropriate target column.

Create Derived Column Mappings

A derived mapping is a mapping between a source column and a target column in which the value of the target column is a function of the source column. For example, you could use a derived column to accomplish the following tasks:

- Write the date to a **Date** field in the target when there is no source column for the date.
- Multiply the value of the **Price** source column by **1.06** to get the value of the **PriceIncludingTax** target column.
- Write the value of the **First Name** and **Last Name** columns in the source table to the **Name** field in the target table.

You can use the techniques illustrated in the following table to create different types of derived column mappings. All of the techniques are used on the **`Mapping`** tab in the properties window for the transformation.

Table 10.1 Derived Column Techniques

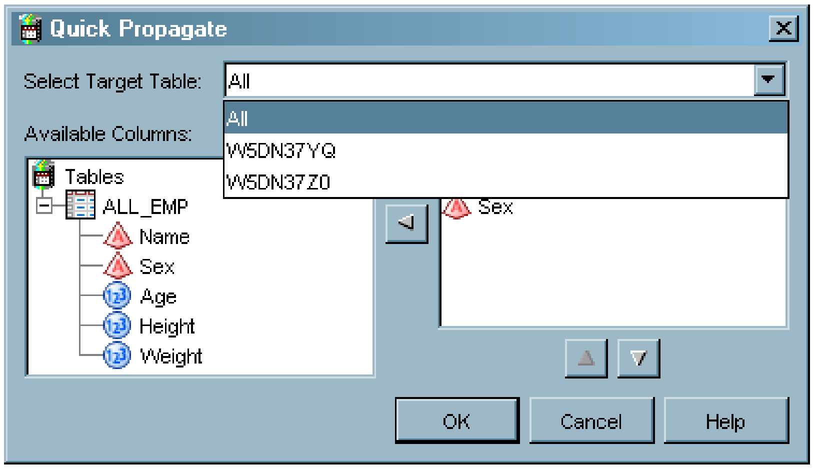
Technique	Description
Directly enter an expression into an **`Expression`** field	You can create any type of expression by entering the expression directly into an **`Expression`** field. The expression can be a constant or an expression that uses the values of one or more source columns. For example, you can create a sample expression that writes today's date to a **`Date`** column in a target table. Perform the following steps: **1** Double-click in the field in which you want to enter the expression. A cursor displays in the field. (The button disappears.) **2** Enter your expression into the field. For example, to write today's date to every row in a column, you can enter the expression &SYSDATE. **3** Click **`OK`**.
Create expressions that use no source columns	Some transformations such as Extract, Lookup, and SCD Type 2 Loader provide an Expression column in the target table. You can perform the following steps to enter an expression into this column that does not use source columns: **1** Right-click in an Expression column. Then, click **`Expression`** in the pop-up menu to access the Expression Builder window. **2** Use the Expression Builder to create an expression. (For information about the Expression Builder window, see the "Expression Builder" topic in SAS Data Integration Studio help.) Then, click **`OK`** to save the expression, close the Expression Builder window, and display the expression in the selected column in the target table. **3** Click **`OK`** to close the properties window for the transformation.

Technique	Description
Create expressions that use a single source column	Assume that you want to define the value of a **`DiscountedPrice`** column in the target by using the **`Price`** source column in an expression. This would be possible if the discount were a constant, such as 6 percent. That is, you might want to define an expression as **`Price * .94`**. You could perform the following steps: **1** Select the Price source column and the DiscountedPrice target column. **2** Right-click either selected variable, and select **`Expression`** from the pop-up menu to access the Expression Builder window. **3** Use the Expression Builder to create an expression. Then, click **`OK`** to save the expression, close the Expression Builder window, and display the expression in the selected column in the target table. **4** Click **`OK`** to close the properties window for the transformation.
Create expressions that use two or more source columns	You can create a derived mapping that uses two or more source columns. Perform the following steps: **1** Select the source columns and target column to be used in the mapping. For example, you could use the values of the **`Price`** and **`Discount`** columns in the source in an expression. Then, the result could be written to the **`DiscountedPrice`** column in the target. **2** Right-click any of the selected variables and select **`New Mapping`** from the pop-up menu that displays. Because two source columns were selected when you requested the new mapping, the warning dialog box is displayed. **3** Click **`Yes`** to access the Expression Builder window. **4** Create the expression, which would be **`Price - (Price * (Discount / 100))`** in this example. Then, click **`OK`** to save the expression, close the Expression Builder window, and display the expression in the selected column in the target table. **5** Click **`OK`** to close the properties window for the transformation.

Create Mappings with Multiple Columns

In certain transformations, the **`Mapping`** tab in the properties window for the transformation displays a single source table and multiple target tables, or multiple source tables and a single target table. To create mappings for multiple sources or targets, you can right-click the **`Mapping`** tab and click **`Quick Propagate`**. The Quick Propagate window is displayed, as depicted in the following display.

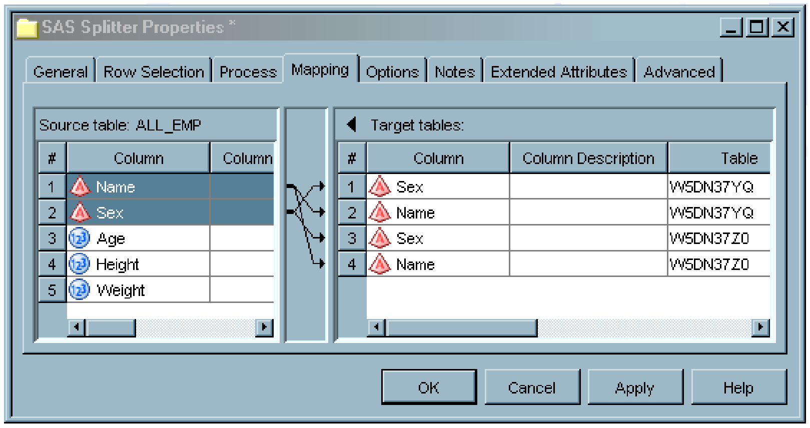
Display 10.2 Quick Propagate Window



You can select **All** in the **Select Target Table** field to propagate the columns listed in the **Selected Columns** field to both of the target tables listed beneath the **All** item. (These tables are moved to the **Selected Columns** field from the **Available Columns** field.

You can also select one of the tables listed in the field to propagate the columns to only that table. In either case, click **OK** to complete the propagation process. The following display shows a **Mapping** tab after two columns are propagated to all of the target tables.

Display 10.3 Source Columns Added to Both Targets



#	Column	Column
1	Name	
2	Sex	
3	Age	
4	Height	
5	Weight	

#	Column	Column Description	Table
1	Sex		W5DN37YQ
2	Name		W5DN37YQ
3	Sex		W5DN37Z0
4	Name		W5DN37Z0

Notice that the Sex and Name columns are added twice, once for the W5DN37YQ target table and once for the W5DN37Z0 target table. If one of the target tables had been selected in the **Select Target Table** field, the Sex and Name columns would appear only once.

Delete Column Mappings

You can delete a column mapping in the **Mapping** tab of the properties window for a transformation by using one of the following methods:

- ☐ Left-click the arrow that connects a column in the **Source table** field to a column in the **Target table** field. Then, press the **Delete** key.

- □ Right-click the arrow that connects a column in the **Source table** field to a column in the **Target table** field. Then, click **Delete** in the pop-up menu.

Note: You must disable automatic mapping for a transformation to delete mappings that would otherwise be automatically created. See "Create Automatic Column Mappings" on page 198. △

Use the Pop-Up Menu Options for Mappings

You can use the pop-up menu in the **Mapping** tab of the properties window to control the behavior of the tab. The available menu options are listed in the following table.

Table 10.2 Pop-Up Options for the Mapping Tab

Menu option	Available when you right-click	Description
Delete	after selecting one or more columns in the **Source table(s)** or **Target table(s)** list box.	Removes one mapping for each selected column. See "Delete Column Mappings" on page 202. To delete the column itself, remove all mappings and press the **Delete** key.
Expression	on a single mapping arrow or in an Expression column in a target table. This menu option is not available in all transformations.	Opens the Expression Builder so that you can manipulate source data before that data is associated with a target table. See "Create Derived Column Mappings" on page 199.
Hide Column	with the cursor in a column heading in the table view.	Removes the specified column from the table view. To restore, use **Show**.
Hold Column to Left	with the cursor in a column heading in a table view.	The selected row moves to the far left and remains there as you scroll left and right.
Hold Column to Right	with the cursor in a column heading in a table view.	The selected column moves to the far right and remains there as you scroll left and right.
Hold Row To Bottom	with the cursor in the # column of a table view.	The selected row moves to the bottom of the table view and remains at the bottom of the table view as you scroll up and down.
Hold Row To Top	with the cursor in the # column of a given row in a table view.	The selected row moves to the top of the table view and remains at the top of the table view as you scroll up and down.
Import Columns	with the cursor in the **Target table(s)** list box.	Opens the Import Columns window, which enables you to define metadata for new columns using existing columns in a current metadata repository.
New Column	with the cursor in the **Target table(s)** list box, with a column optionally selected to set the insertion point.	Adds a new column below the selected column or at the bottom of the table. The name of the new table is **Untitled***n*, where *n* is a number that is assigned automatically. Double-click on the default name to enter a new name.
New Mapping	after selecting a source column with a left-click and selecting a target column with a Shift-left-click.	Adds a mapping arrow between a source column and a target column.

Menu option	**Available when you right-click**	**Description**
Quick Map	with the cursor anywhere in the **Mapping** tab.	Automatically adds mappings between source and target columns that have the same name, data type (numeric or character), and length. See "Create Automatic Column Mappings" on page 198
Quick Propagate	with the cursor anywhere in the **Mapping** tab.	Opens the Quick Propagate window, which enables you to define new target columns and add mappings to those new target columns based on existing source columns. See "Create Mappings with Multiple Columns" on page 201.
Release	with the cursor in the # column or in a column heading.	Enables you to regain the ability to reorder any rows that have been held to the left or right, or any columns that have been held to the top or bottom.
Select All Mappings	with the cursor anywhere in the **Mapping** tab.	Selects all mappings, generally for deletion.
Show	with the cursor in a column heading.	Displays any or all columns in a table view that have been hidden with the **Hide Columns** menu option.
Sort Ascending	with the cursor in a column heading.	Sorts all rows in ascending alphanumeric order based on the values in the selected column.
Sort Descending	with the cursor in a column heading.	Sorts rows in descending alphanumeric order based on the values in the selected column. Rows that are held to the top or bottom are not included in the sort.
Sort Original	with the cursor in a column heading.	Restores the column sort order to the order that is displayed when the Properties window is first displayed. Any prior ascending or descending sort order is removed.

About Archived Transformations

In order to support backwards compatibility for existing processes and guarantee that processes run exactly as defined using older transformations, SAS has developed a methodology for archiving older versions of transforms in the Process library. The process library will continue to surface the archived transformations for some number of releases. When a job is opened that contains a newer transformation replacement, a dialog box displays that indicates the name of the old transformation. The dialog box also provides the name and location of the new transformation in the process library tree.

The following transformations are being archived and replaced with newer versions in version 3.4 of SAS Data Integration Studio:

- Table Loader
- SQL Join
- Fact Table

In addition, older transformations will be marked with a flag on their icons. This flag indicates that each transformation is an older version of an updated transformation.

CHAPTER 11

Working with Generated Code

About Code Generated for Jobs

Overview

When SAS Data Integration Studio generates code for a job, it typically generates the following items:

- specific code to perform the transformations used in the job

- a LIBNAME statement for each table in the job
- a SYSLAST macro statement at the end of each transformation in the job
- remote connection statements for any remote execution machine that is specified in the metadata for a transformation within a job
- macro variables for status handling

You can set options for the code that SAS Data Integration Studio generates for jobs and transformations. For details, see “Specifying Options for Jobs” on page 211 and “Specifying Options for a Transformation” on page 211.

LIBNAME Statements

When SAS Data Integration Studio generates code for a job, a library is considered *local* or *remote* in relation to the SAS Application Server that executes the job. If the library is stored on one of the machines that is specified in the metadata for the SAS Application Server that executes the job, it is local. Otherwise, it is remote.

SAS Data Integration Studio generates the appropriate LIBNAME statements for local and remote libraries.

Here is the syntax that is generated for a local library:

```
libname libref <enginer> <"lib-specification"> <connectionOptions>
<libraryOptions> <schema=databaseSchema> <user=userID> <password=password>;
```

Here is the syntax that is generated for a remote library:

```
options comamid=connection_type;
%let remote_session_id=host_name <host_port>;
signon remote_session_id <user=userID> <password=password>;
rsubmit remote_session_id;
   libname libref <enginer> <"lib-specification"> <connectionOptions>
   <libraryOptions> <schema=databaseSchema> <user=userID> <password=password>;
endrsubmit;

rsubmit remote_session_id;
proc download data=table_on_remote_machine
out=table_on_local_machine;
run;
endrsubmit;
```

SYSLAST Macro Statements

The **Options** tab in the property window for most transformations includes a field that is named **Create SYSLAST Macro Variable**. This field specifies whether SAS Data Integration Studio generates a SYSLAST macro statement at the end of the current transformation. In general, accept the default value of **YES** for the Create SYSLAST Macro Variable option.

Remote Connection Statements

Each transformation within a job can specify its own execution host. When SAS Data Integration Studio generates code for a job, a host is considered local or remote in relation to the SAS Application Server that executes the job. If the host is one of the machines that is specified in the metadata for the SAS Application Server that executes the job, it is local. Otherwise, it is remote.

A remote connection statement is generated if a remote machine has been specified as the execution host for a transformation within a job:

```
options comamid=connection_type;
%let remote_session_id=host_name <host_port>;
signon remote_session_id <user=userID password=password>;
rsubmit remote_session_id;
... SAS code ...
endrsubmit;
```

Note: This will be done implicitly for users if the machine is remote. Users can also use the DataTransfer transformation to explicitly handle moving data between machine environments when needed. The Data Transfer transformation provides for more control over the transfer when needed, such as support for locale-specific settings. △

Macro Variables for Status Handling

When SAS Data Integration Studio generates the code for a job, the code includes a number of macro variables for status handling. For details, see "About Monitoring Jobs" on page 157.

User Credentials in Generated Code

The code that is generated is based on the credentials and permission settings of the user that generated the code. When required, such as in LIBNAME statements to a relational DBMS, for passthrough, or for remote machine data movement, the generated code might also contain embedded credentials, with encoded passwords.

If the credentials of the person who created the job are changed and a deployed job contains outdated user credentials, the deployed job fails to execute. The solution is to redeploy the job with the appropriate credentials. For details, see "Redeploying Jobs for Scheduling" on page 173.

Displaying the Code Generated for a Job

Problem

You want to see the code that you generated for a job.

Solution

SAS Data Integration Studio uses the metadata in a job to generate code or to retrieve user-written code. You can display the SAS code for a job by opening the job in the Process Designer window and selecting the **Source Editor** tab.

Tasks

This section describes two ways to display the SAS code for a job. It is assumed that you are working under change management. However, you do not have to check out a job to generate and display its code.

Prerequisite

SAS Data Integration Studio must be able to connect to a SAS Application Server with a SAS Workspace Server component in order to generate the SAS code for a job.

View Code Displayed in the Process Designer Window

To view the code for a job that is currently displayed in the Process Designer window, click the `Source Editor` tab. The code for the job is displayed in the Source Editor.

View Code Not Displayed in the Process Designer Window

Perform the following steps to view the code for a job that is not displayed in the Process Designer window:

1 From the SAS Data Integration Studio desktop, in the Inventory tree, expand the `Jobs` folder.

2 Do one of the following:

- □ Select the job, go to the `View` drop-down menu, and select `View Code`.
- □ Right-click the job that you want to view, then select `View Code` from the pop-up menu.

  The code for the job is displayed in the Source Editor.

Displaying the Code Generated for a Transformation

Problem

You want to see the code that you generated for a transformation.

Solution

To see the code as part of the job, see “Displaying the SAS Code for a Job” on page 152.

To see the code generated specifically for a transformation, see the following tasks section.

Tasks

Perform the following steps to see the code generated for a transformation:

1 Right-click on the transformation icon.

2 Select `View Step Code` from the pop-up menu.

A new Source Editor window will open. The code generated for the transformation will be in this window.

Specifying Options for Jobs

Problem

You want to set options for Data Integration Studio jobs, such as enabling parallel processing and configuring grid processing.

In most cases the appropriate options are selected by default. You can override the defaults by using this procedure.

Solution

You can set global options for jobs on the **`Code Generation`** tab of the **`Options`** menu. The **`Options`** menu is available on the **`Tools`** menu on the SAS Data Integration Studio menu bar. You can set local options on the **`Options`** tab available on the properties window for each job.

Tasks

Set Global Options for Job

You can set the options listed on the **`Code Generation`** tab of the **`Options`** menu available on the **`Tools`** menu on the SAS Data Integration Studio menu bar. The options that are relevant for jobs are examined in the table Table 7.1 on page 146.

Set Local Options for Jobs

You can set local options that apply to individual jobs by selecting the job and using the right mouse button to open the pop-up menu. Select **`Properties`** and then select the **`Options`** tab. These local options override global options for the selected job, but they do not affect any other jobs. For the local options, refer to Table 7.2 on page 147.

You can also add code to a job using the **`Pre and Post Process`** option tab available on the properties window for each job. Select the check box for **`PreProcessing`** and select the **`Edit`** button. The Edit Source Code window enables you to add or update user-written source code.

Specifying Options for a Transformation

Problem

You want to set options for a Data Integration Studio transformation, such as SAS Sort, SQL Join, or Extract.

Solution

You can specify SAS system options, SAS statement options, or transformation-specific options on the **Options** tab or other tabs in the properties window for many transformations. Use this method to select these options when a particular transformation executes.

Tasks

Perform the following steps to display the **Options** tab in the properties window for a transformation in a job:

1 Open the job to display its process flow.
2 Right-click the transformation and select **Properties** from the pop-up menu.
3 Select the **Options** tab.

For a description of the available options for a particular transformation, see the Help for the **Options** tab or other tabs that enable you to specify options. If the **Options** tab includes a **System Options** field, you can specify options such as UBUFNO for the current transformation. Some transformations enable you to specify options that are specific to that transformation. For example, the **Options** tab for the Sort transformation has specific fields for sort size and sort sequence. It also has a **PROC SORT Options** field where you can specify sort-related options that are not otherwise surfaced in the interface. For example, you could use these fields to specify the options that are described in the "Setting Sort Options" on page 276.

In addition to the **Options** tab, some transformations have other tabs that enable you to specify options that affect performance. For example, the properties window for the SAS Scalable Performance Server Loader transformation has an **Other SPD Server Settings** tab, which enables you to specify several SAS Scalable Performance Server options.

Modifying Configuration Files or SAS Start Commands for Application Servers

There are several ways to customize the environment where the code generated by Data Integration Studio runs.

When you submit a SAS Data Integration Studio job for execution, it is submitted to a SAS Workspace Server component of the relevant SAS Application Server. The relevant SAS Application Server is one of the following:

- the default server that is specified on the **SAS Server** tab in the Options window
- the SAS Application Server to which a job is deployed with the Deploy for Scheduling option

To set SAS invocation options for all SAS Data Integration Studio jobs that are executed by a particular SAS server, specify the options in the configuration files for the relevant SAS Workspace Servers, batch or scheduling servers, and grid servers. (You do not set these options on SAS Metadata Servers or SAS Stored Process Servers.) Examples of these options include UTILLOC, NOWORKINIT, or ETLS_DEBUG.

To specify SAS system options or startup options for all jobs that are executed on a particular SAS Workspace Server, modify one of the following for the server:

- □ config.sas file
- □ autoexec.sas file
- □ SAS start command

For example, your SAS logs have become too large and you want to suppress the MPRINT option in your production environment. Invoke the ETLS_DEBUG option in the autoexec.sas file by following these steps:

1. Open the autoexec.sas file.
2. Add the following code to the autoexec.sas file for your production run:

   ```
   %let etls_debug=0;
   ```

3. Save and close the file.

Note: If the condition **`etls_debug=0`** is true, then the logic in the deployed job prevents execution of the **`OPTIONS MPRINT;`** statement. To turn on the MPRINT option again, remove **`%let etls_debug=0;`** from the autoexec.sas file. △

CAUTION:
We strongly recommend that you do not turn off MPRINT in a development environment. △

CHAPTER
12

Working with User-Written Code

About User-Written Code

A job is collection of SAS tasks that create output. By default, SAS Data Integration Studio uses the metadata for a job to generate code for the job. You can also specify user-written code for the entire job or for any transformation within the job.

If you want to supply user-written code for a job or a transformation within a job, you create the code and then specify the location of the user-written code in the metadata for the job or transformation. SAS Data Integration Studio then retrieves the specified code instead of generating code.

In most cases, you can specify user-written code for a job or a transformation by displaying the properties window for the job or the properties window for the transformation and then updating the metadata on the **Process** tab. Cubes are the only exception to this rule. The job for a cube can be updated only through the Cube Designer, not the job properties window. For details about specifying user-written code for cubes, see the "User-Written Source Code for Cubes" topic in SAS Data Integration Studio help.

For details about adding status handling macros to the user-written code for a job or transformation, see the "Macro Variables for Status Handling" topic in SAS Data Integration Studio help. The following table describes the main methods for working with user-written code in jobs and transformations.

Table 12.1 Methods for Working With User-Written code

If you want to	**Do this**
Specify user-written code in a transformation	Specify user-written code for a transformation within a job in one of the following ways:. □ For an existing transformation, generate code for the transformation, edit the generated code and save the edited code, and then specify the location of the edited code in the metadata for the transformation. □ For an existing transformation, write a SAS program that performs the transformation that you want and then specify the location of the user-written code in the metadata for the transformation. □ For a new transformation (one that does not currently exist in the job process flow diagram), write a SAS program that performs the transformation that you want, add a User Written Code transformation to the process flow diagram, and then specify the location of the user-written code in the metadata for the User Written Code transformation. For details, see "Editing the Generated Code for a Transformation" on page 221.
Specify user-written code for an entire job	Specify user-written code for an entire job in one of the following ways: □ For an existing job, generate code for the job; edit the generated code and save the edited code; then specify the location of the edited code in the metadata for the job. □ For an existing job, write a SAS program that performs the desired job and then specify the location of the user-written code in the metadata for the job. □ For a new job, write a SAS program that performs the desired job and then specify the location of the user-written code in the metadata for the job. For details, see the following topics: "Editing the Generated Code for a Job" on page 218. "Adding User-Written Source Code to an Existing Job" on page 219. "Creating a New Job That Consists of User-Written Code" on page 220.

If you want to	Do this
Create user-written transformations	The most common way to create a custom transformation that you can drag from the Process Library and drop into any job is to create a generated transformation that uses SAS code to produce the output that you want. In SAS Data Integration Studio, use the Transformation Generator wizard to include your SAS code in a transformation, which is then available in the Process Library. For details, see "Creating and Using a Generated Transformation" on page 226.
Use the User Written Code transformation	The User Written Code transformation is provided by default in the Process Library tree. It enables you to add user-written code to a job. For details, see "About the User Written Code Transformation" on page 222.

Editing the Generated Code for a Job

Problem

You want to modify the code that is generated for a job in order to get the desired output from the job.

Solution

When you work with an existing job, you can generate SAS code for the job. Then, you can edit the generated code and save the edited code. Finally, you need to specify the location of the edited code in the metadata for the job. This edited code is saved as a file on the file system. Therefore, you might want to create a special directory for this type of code. Naturally, this method requires a basic understanding of the SAS programming language.

Tasks

Edit and Save the Generated Code

Perform the following steps to generate code for a job, edit the code, and then save the edited code to a file:

1 Right-click a job and click **`View Job`** in the pop-up menu. The job opens in the Process Editor.

2 Click the **`Source Editor`** tab in the Process Editor. Code is generated for the entire job and displayed in the **`Source Editor`** tab.

3 Edit the generated code in the **`Source Editor`** tab. Then, close the Process Designer window. Click **`Yes`** when prompted to save your changes. A file selection window displays.

4 Specify a path name for the edited code in the file selection window. Then click **`Save`**. The edited code is saved to a file, and the Process Designer window closes.

At this point, you have saved the edited code. The next step is to specify the location of the edited code in the metadata for the job.

Specify a Path to the Edited Code in the Job Metadata

Perform the following steps to specify a path to the edited code in the job metadata:

1 Open the **`Process`** tab in the properties window for the job.

2 Click **`User Written`**. The **`Type`** field and related fields become active.

3 Select **`File`** with the drop-down menu in the **`Type`** field.

4 You will usually accept the default host in the **`Host`** field.

5 Specify a path to the file that contains the edited code in the **`Path`** field. The server in the **`Host`** field must be able to resolve this path. You can type in the path or use the **`Browse`** button to display a file selection window.

6 After specifying the path, click **`OK`** to save your changes.

The specified user-written code is retrieved whenever code for this job is generated. After you have specified user-written code in the metadata for a job, you might want to execute the job to verify that it works in the context of SAS Data Integration Studio.

Adding User-Written Source Code to an Existing Job

Problem

You want to completely replace the code that is generated for a job in order to get the desired output from the job.

Solution

You can use the **`Process`** tab in the properties window for the job to write your own SAS code for an existing job. Then, you can specify the location of your code in the metadata for the job.

Tasks

Write SAS Code and Add It to An Existing Job

Perform the following steps to write the code and add it to an existing job:

1 Write a SAS program that will perform the desired job. Verify that your program produces the appropriate output.

2 Open the **`Process`** tab in the properties window for the job.

3 Click **`User Written`** in the **`Code Generation`** group box. The **`Type`**, **`Name`**, and **`Description`** fields become active.

4 You can now perform one of the following steps to specify the location of the user-written SAS program:

   □ Select **`File`** in the drop-down menu in the **`Type`** field if you want to maintain the SAS program as a file on the file system. Then select the server that you use to access the file in the **Host** field. Finally, specify the physical path to the SAS program in the **Path** field.

- Select **Metadata** in the drop-down menu in the **Type** field if you want to maintain the SAS program as an object in the current metadata repository. Begin by clicking **Edit** to display the Edit Source Code window. Next, copy the SAS program and paste it into the window. Then, click **OK** to save the code to the current metadata repository. Finally, enter a name for the metadata object that specifies the location of the user-written code in the **Name** field. You can also enter a **Description**.

5 Accept the default SAS Application Server or select a different application server in the **Execution Host** field. This server executes the user-written source code.

6 Click **OK** to save your changes.

The specified user-written code is retrieved whenever code for this job is generated. After you have specified user-written code in the metadata for a job, you might want to execute the job to verify that it works in the context of SAS Data Integration Studio.

Creating a New Job That Consists of User-Written Code

Problem

You want to create a new job that consists entirely of user-written code.

Solution

You can write a SAS program that produces the desired output. Then, you can add metadata for the job and specify the location of the user-written code. For documentation purposes, you can add a process flow diagram to the metadata for the job, but a process flow diagram is optional when user-written code is supplied for the entire job.

Tasks

Create a Job That Consists of User-Written Code

Perform the following steps to write the code and use it to create the job:

1 Write a SAS program that will perform the desired job. Verify that your program produces the appropriate output.

2 Create a new job and give it an appropriate name. The Process Designer window for the new job is displayed.

3 Open the **Process** tab in the properties window for the job.

4 Click **User Written** in the **Code Generation** group box. The **Type**, **Name**, and **Description** fields become active.

5 You can now perform one of the following steps to specify the location of the user-written SAS program:

- ☐ Select **`File`** in the drop-down menu in the **`Type`** field if you want to maintain the SAS program as a file on the file system. Then select the server that you use to access the file in the **`Host`** field. Finally, specify the physical path to the SAS program in the **`Path`** field.
- ☐ Select **`Metadata`** in the drop-down menu in the **`Type`** field if you want to maintain the SAS program as an object in the current metadata repository. Begin by clicking **`Edit`** to display the Edit Source Code window. Next, copy the SAS program and paste it into the window. Then, click **`OK`** to save the code to the current metadata repository. Finally, enter a name for the metadata object that specifies the location of the user-written code in the **`Name`** field. You can also enter a **`Description`**.

6 Accept the default SAS Application Server or select a different application server in the **`Execution Host`** field. This server executes the user-written source code.

7 Click **`OK`** to save your changes.

The specified user-written code is retrieved whenever code for this job is generated. After you have specified user-written code in the metadata for a job, you might want to execute the job to verify that it works in the context of SAS Data Integration Studio.

Editing the Generated Code for a Transformation

Problem

You want to modify the code that is generated for a particular transformation in order to get the desired output from the transformation.

Solution

When you work with an existing transformation in a process flow diagram, you can generate SAS code for the transformation. Then, you can edit the generated code and save the edited code. Finally, you need to specify the location of the edited code in the metadata for the transformation. This edited code is saved as a file on the file system. Therefore, you might want to create a special directory for this type of code. Naturally, this method requires a basic understanding of the SAS programming language.

Tasks

Edit and Save the Generated Code

Perform the following steps to generate code for a transformation, edit the code, and then save the edited code to a file:

1 Right-click the job that contains the transformation that you want and click **`View Job`** in the pop-up menu. The job opens in the Process Editor window.

2 Right-click the transformation and click **`View Step Code`**. Click the **`Source Editor`** tab in the Process Editor window. Code is generated for the transformation and displayed in the Source Editor window.

3 Edit the generated code in the Source Editor window. Then, close the window. Click **Yes** when prompted to save your changes. A file selection window displays.
4 Specify a pathname for the edited code in the file selection window. Then click **Save**. The edited code is saved to a file, and the Source Editor window closes.

At this point, you have saved the edited code. The next step is to specify the location of the edited code in the metadata for the transformation.

Specify a Path to the Edited Code in the Transformation Metadata

Perform the following steps to specify a path to the edited code in the transformation metadata:

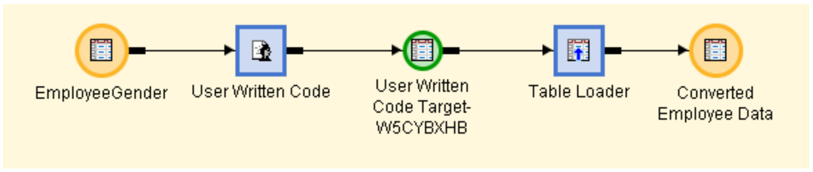
1 Open the **Process** tab in the properties window for the transformation.
2 Click **User Written**. The **Type** field and related fields become active.
3 Select **File** with the drop-down menu in the **Type** field.
4 You will usually accept the default host in the **Host** field.
5 Specify a path to the file that contains the edited code in the **Path** field. The server in the Host field must be able to resolve this path. You can type in the path or use the **Browse** button to display a file selection window.
6 After specifying the path, click **OK** to save your changes.

The specified user-written code is retrieved whenever code for this transformation is generated. After you have specified user-written code in the metadata for a transformation, you might want to execute the job that contains the transformation to verify that it works in the context of SAS Data Integration Studio.

About the User Written Code Transformation

When you create a process flow diagram for a job, you can drag and drop transformations from the Process Library tree into the Process Editor. Sometimes, however, the predefined transformations in the Process Library tree might not meet your needs. In this cases, you can write a SAS program that produces the desired output. Then, you can add your code to a User Written Code transformation in the process flow diagram. Finally, you can specify the location of the new code in the metadata for the User Written Code transformation. The following display shows a sample User Written Code transformation dropped into a process flow diagram.

Display 12.1 Sample User Written Code Transformation In a Job



After the transformation has been inserted, you can update its metadata so that it specifies the location of your user-written code. When the job is executed, the user-written code is retrieved and executed as part of the job.

Note: When SAS Data Integration Studio generates all of the code for a job, it can automatically generate the metadata for column mappings between sources and targets. However, when you specify user-written code for part of a job, you must manually define the column metadata for that part of the job that the user-written code handles. SAS Data Integration Studio needs this metadata to generate the code for the part of the job that comes after the User Written Code transformation.

For the example depicted in the process flow diagram in the display, you need to manually define the column metadata in the User Written Code transformation and the Loader transformation to create the target, Converted Employee Data. For details, see "Creating a Job That Includes the User Written Code Transformation" on page 223.

Note that you can add the RCSET macro and the TRANS_RC and JOB_RC variables to user-written code, such as the code for User Written transformations and generated transformation templates. △

Creating a Job That Includes the User Written Code Transformation

Problem

You want to add user-written code to a job. One method uses the User Written Code transformation, which is provided by default in the SAS Data Integration Studio Process Library. Once you place this transformation in a job, you can add user-written code in the **Process** tab of its properties window and map its columns to the target table. This approach works particularly well with jobs that need quick custom code or require only one input and output and no parameters. More complicated situations are handled more effectively with the Transformation Generator wizard. For more information, see "Creating and Using a Generated Transformation" on page 226.

Solution

You can create a job that includes the User Written Code transformation. You need to add the code to the job in the User Written Code transformation. Then, you must map the columns from the source table to the work table. Finally, you need to map the columns from the work table to the target table.

Tasks

Add User-Written Code to a Job

Perform the following steps to add the user-written code to a job:

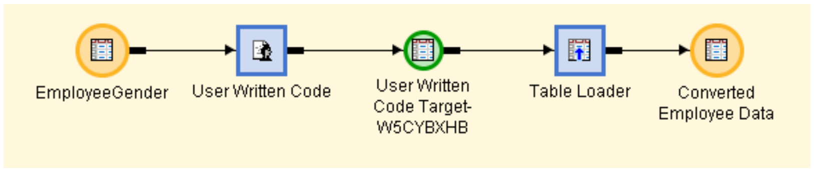
1 Write SAS code and test it to ensure that it produces the required output. The following code was written for the sample job:

```
data &_OUTPUT;
  set &SYSLAST;
  length sex $1;
  if gender = 1 then
    sex = "M";
  else if gender = 2 then
```

```
        sex = "F";
      else
      sex="U";
    run;
```

2 Create a new job and give it an appropriate name. The Process Designer window for the new job is displayed.
3 Drop the User Written Code transformation from the Data Transformations folder in the Process Library tree into the empty job.
4 Drop the source table on the drop zone for the User Written Code transformation.
5 Drop the target table on the target drop zone for the User Written Code transformation. The flow for the sample job is shown in the following display.

Display 12.2 Sample User Written Code Transformation In a Job



Note that a work table and the Table Loader transformation are automatically included in the job.

6 Open the **Process** tab in the properties window for the User Written Code transformation.
7 Click **User Written** in the **Code Generation** group box. The **Type**, **Name**, and **Description** fields become active.
8 Select **Metadata** in the drop-down menu in the **Type** field.
9 Click **Edit** to display the Edit Source Code window. Then, copy the SAS program and paste it into the window.
10 Click **OK** to save the code to the current metadata repository. Enter a name for the metadata object that specifies the location of the user-written code in the **Name** field. You can also enter a **Description**.
11 Accept the default SAS Application Server or select a different application server in the **Execution Host** field. This server executes the user-written source code.
12 Click **Apply** to save your changes. Do not close the properties window for the User Written Code transformation.

At this point, you have updated the User Written Code transformation so that it can retrieve the appropriate code when the job is executed.

Map the Columns From the Source Table to the Work Table

When SAS Data Integration Studio generates all of the code for a job, it can automatically generate the metadata for column mappings between sources and targets. However, when you use user-written code for part of a job, you must manually define the column metadata for that part of the job that the user-written code handles. SAS Data Integration Studio needs this metadata to generate the code for the part of the job that comes after the User Written Code transformation. Perform the following steps to map between the columns in the source table and the columns in the work table:

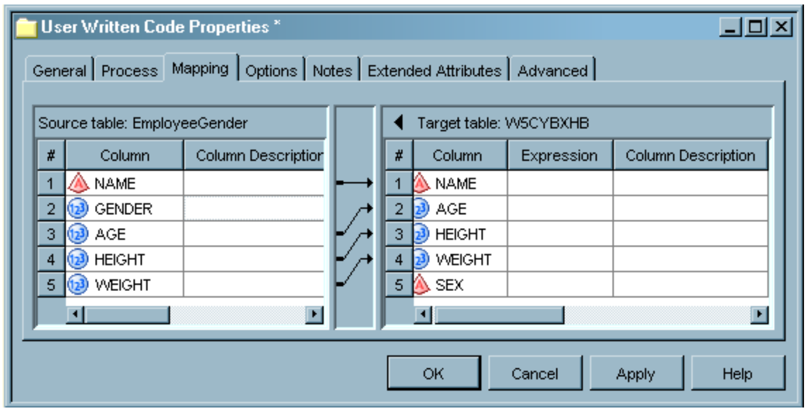
1 Open the **Mapping** tab in the properties window for the User Written Code transformation.

2 Review the mappings between the source table in the **`Source table`** field and the work table in the **`Target table`** field. In the sample job, the following steps were required to map the columns properly:

   a Right-click the GENDER column in the **`Target table`** field. Then, click **`Delete Columns`**.

   b Right-click in the **`Target table`** field and click **`New Column`**. Give the new column an appropriate name (SEX, in this case). You should also make sure that column properties such as type and length match the properties of same row in the target table. In this case, the length of the column needed to be changed to **`1`**. The mappings for this part of the sample job are shown in the following display.

Display 12.3 Sample Mapping From Source Table to Work Table



Source table: EmployeeGender

#	Column	Column Description
1	NAME	
2	GENDER	
3	AGE	
4	HEIGHT	
5	WEIGHT	

Target table: W5CYBXHB

#	Column	Expression	Column Description
1	NAME		
2	AGE		
3	HEIGHT		
4	WEIGHT		
5	SEX		

   c Click **`OK`** to save the settings and close the properties window for the User Written Code transformation.

3 Click **`OK`** to save the settings and close the properties window for the User Written Code transformation.

Map the Columns From the Work Table to the Target Table

Perform the following steps to map between the columns in the source table and the columns in the work table:

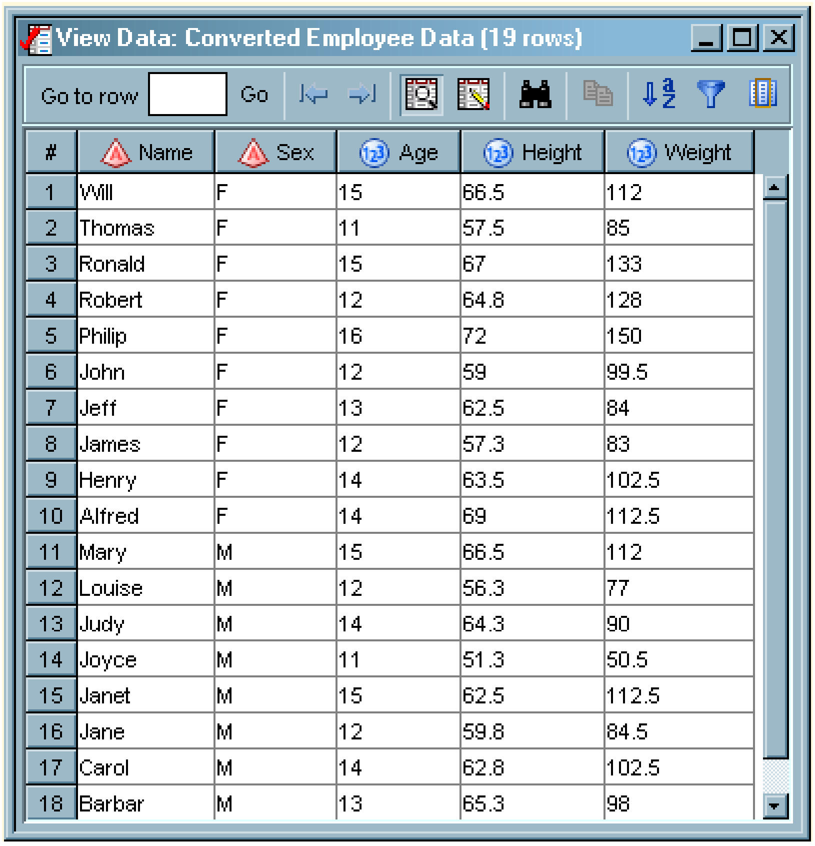
1 Open the **`Mapping`** tab in the properties window for the Table Loader transformation.

2 Review the mappings between the work table in the **`Source table`** field and the target table in the **`Target table`** field. In the sample job, the SEX column in the **`Source table`** field was mapped to the SEX column in the **`Target table`** field.

3 Click **`OK`** to save the settings and close the properties window for the Table Loader transformation.

4 Run the job. The output of job is shown in the following display.

Display 12.4 Output From the Sample Job



View Data: Converted Employee Data (19 rows)

#	Name	Sex	Age	Height	Weight
1	Will	F	15	66.5	112
2	Thomas	F	11	57.5	85
3	Ronald	F	15	67	133
4	Robert	F	12	64.8	128
5	Philip	F	16	72	150
6	John	F	12	59	99.5
7	Jeff	F	13	62.5	84
8	James	F	12	57.3	83
9	Henry	F	14	63.5	102.5
10	Alfred	F	14	69	112.5
11	Mary	M	15	66.5	112
12	Louise	M	12	56.3	77
13	Judy	M	14	64.3	90
14	Joyce	M	11	51.3	50.5
15	Janet	M	15	62.5	112.5
16	Jane	M	12	59.8	84.5
17	Carol	M	14	62.8	102.5
18	Barbar	M	13	65.3	98

Creating and Using a Generated Transformation

Problem

You want to create a generated transformation that will enable you to process multiple outputs or inputs, macro variables, and parameters. Then, you need to use the generated transformation in a SAS Data Integration Studio job. Once written and saved, the generated transformation can be used in any job.

Solution

One of the easiest ways to customize SAS Data Integration Studio is to write your own generated transformations. The Transformation Generator wizard in SAS Data Integration Studio guides you through the steps of specifying SAS code for the transformation and saving the transformation in a current metadata repository. After the transformation is saved and checked in, it displays in the Process Library tree, where it is available for use in any job.

Tasks

Write a Generated Transformation

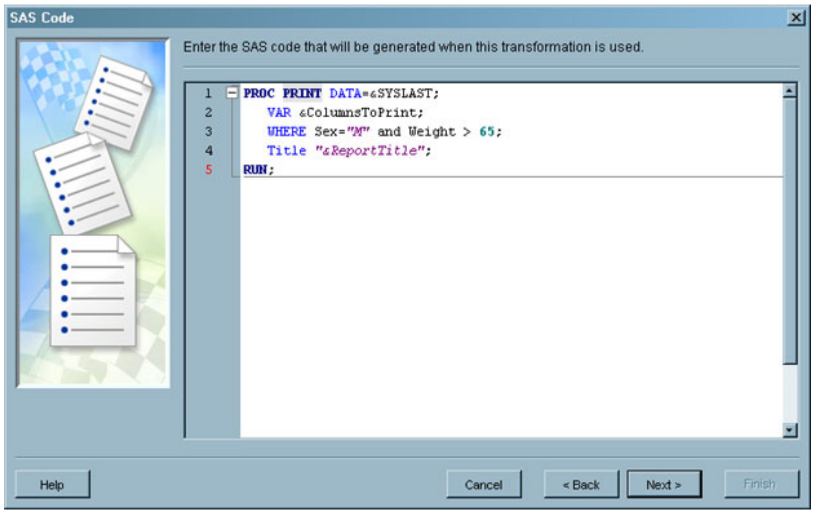
Perform the following steps to write a generated transformation:

1 Open the Transformation Generator wizard from the **`Tools`** menu in the SAS Data Integration Studio menu bar. The wizard opens to the general information screen.

2 Enter an appropriate name for the transformation. Then, select the subfolder beneath the Process Library folder where you would like to place the transformation from the drop-down menu in the **`Process Library Folder`** field. You can also enter a description of the transformation. Click **`Next`** to access the SAS Code screen.

3 Enter the SAS code that will be generated by the transformation. Of course, you can either enter code manually or paste in SAS code from an existing source. The following display shows the SAS code for a sample generated transformation.

Display 12.5 Sample Transformation Code

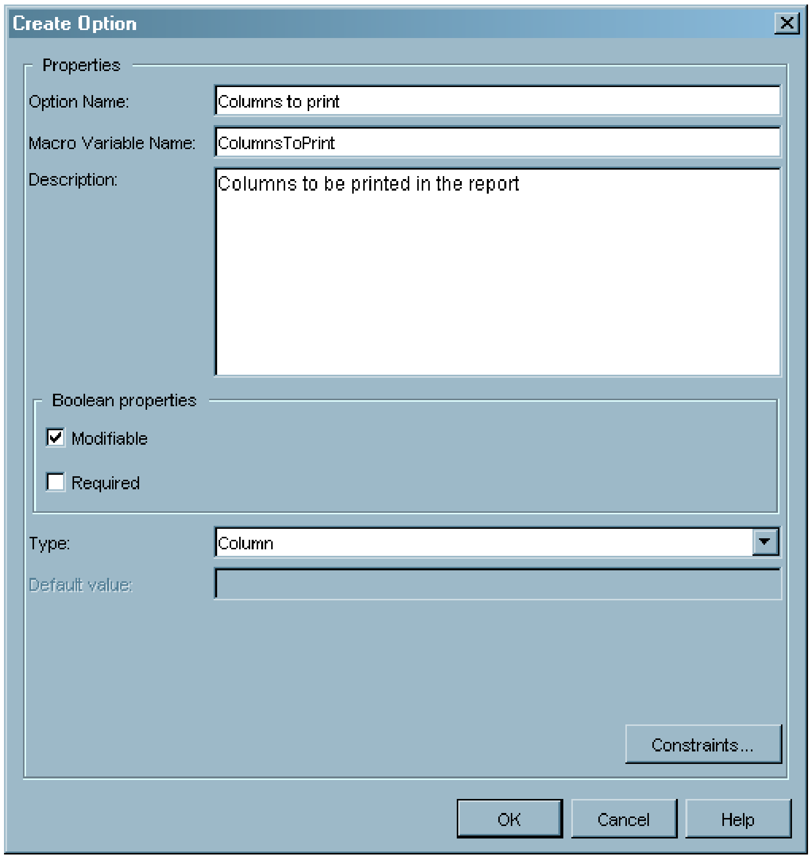

SAS Code
Enter the SAS code that will be generated when this transformation is used.
Help
Cancel
< Back
Next >
Finish

```
PROC PRINT DATA=&SYSLAST;
   VAR &ColumnsToPrint;
   WHERE Sex="M" and Weight > 65;
   Title "&ReportTitle";
RUN;
```

A number of macro variables appear in this sample code. One of these, &SYSLAST, is normally available and refers to the last data set created. The transformation also includes other macro variables, such as &ColumnsToPrint. The type of each such variable is defined in the Options screen of the wizard. You will supply values for these user-defined variables when the transformation is included in a job. Click **`Next`** to access the Options window.

4 Click **`New`** to access the Create Option window. Define an option that corresponds to the first macro variable listed on the SAS code screen. The following display shows a Create Option window for the sample transformation.

Display 12.6 Sample Create Options Window



Option Name: Columns to print

Macro Variable Name: ColumnsToPrint

Description: Columns to be printed in the report

[✓] Modifiable

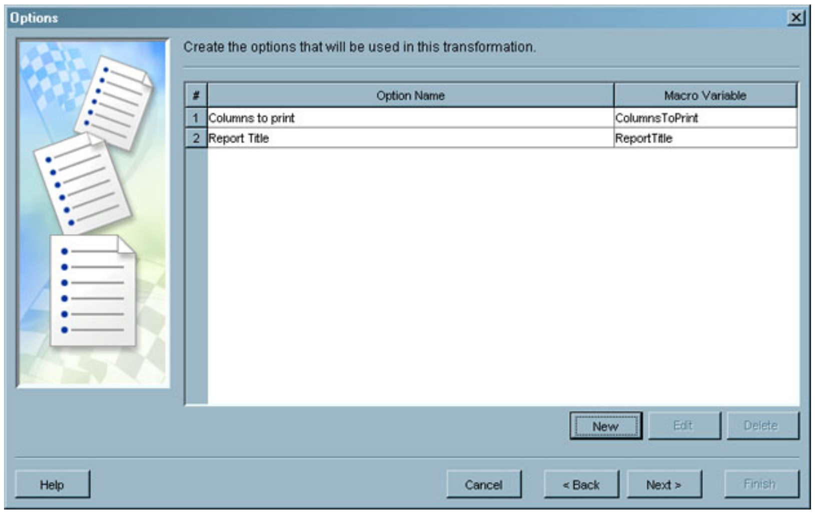
[ ] Required

Type: Column

Default value:

Note that you provide an option name, a macro name, a description, and a data type for the option. If there are constraints on an option, you can click **`Constraints`** to access the Constraints window. The constraints are different for each data type. You need to define each of the macro variables included in the transformation as an option. These options display on the **`Options`** tab of the transformation when it is used in a job. The completed Options screen for the sample transformation is depicted in the following display.

Display 12.7 Completed Options Screen

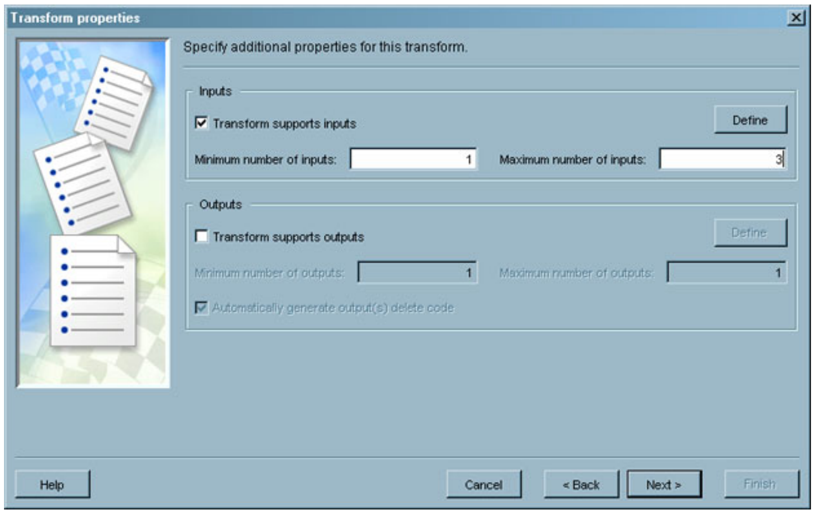

Options
Create the options that will be used in this transformation.
New
Edit
Delete
Help
Cancel
< Back
Next >
Finish

#	Option Name	Macro Variable
1	Columns to print	ColumnsToPrint
2	Report Title	ReportTitle

When you have defined options for each of the macro variables, click **`Next`** to access the Transform properties screen.

5 Use the Transform properties screen to specify the number of inputs and outputs for the generated transformation. The Transform properties screen for the sample transformation is depicted in the following display.

Display 12.8 Transform Properties



☑ Transform supports inputs

Minimum number of inputs: 1 Maximum number of inputs: 3

☐ Transform supports outputs

Minimum number of outputs: 1 Maximum number of outputs: 1

☑ Automatically generate output(s) delete code

These values add a single input drop zone to the transformation when it is used in a job. The number of drop zones displayed in the Process Designer window is equal to the value in the **`Minimum number of inputs`** field. Therefore, only one drop zone is displayed. If you later update the transformation to increase this minimum number of inputs value, any jobs that have been submitted and saved use the original value. The increased minimum number of inputs is enforced only for subsequent jobs. This means that you can increase the minimum number of inputs without breaking existing jobs.

The increased maximum number of inputs is used to allow you to drop additional inputs into the drop zone. (In the sample transformation, you can have up to three inputs because you set the maximum to three.) The same rules apply to outputs. The report generated by this transformation is sent to the **Output** panel of the Process Designer window. Therefore, you do not need to add an output drop zone to the transformation by using the controls in the **Outputs** group box.

6 After you have set up these properties, click **Next** to access the Select Folder window. Select an appropriate Custom tree folder for this generated transformation.

7 After you have selected a Custom tree folder, click **Next** to access the Wizard Finish window. Verify that the metadata is correct, and then click **Finish**. Your transformation is created and saved.

8 Verify that the generated transformation is available in the Process Library. From the SAS Data Integration Studio desktop, click the tab for the Process Library tree. Locate the transformation that you just created.

Use a Generated Transformation in a Job

Perform the following steps to create and run a job that contains the generated transformation:

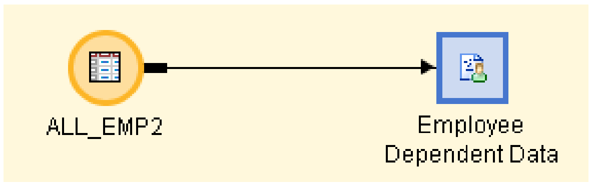
1 Create an empty job.

2 Drag the generated transformation from its folder in the Process Library tree. Then, drop it into the empty job.

3 Drop the source table for the job into the input drop zone for the generated transformation.

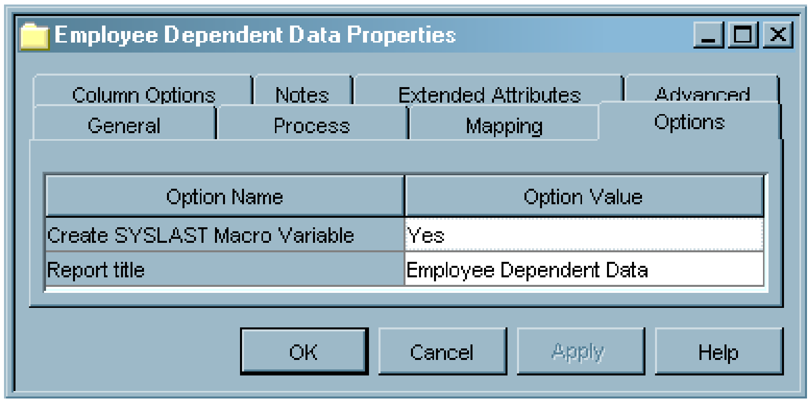
4 Drop the target table into the output drop zone for the generated transformation, if an output exists. You can also send the output to the **Output** tab of the Process Designer window. The appropriate option must be set so that the **Output** tab appears in the Process Designer window. For details, see the "Process Designer Window" topic in SAS Data Integration Studio help. The sample job shown in the following display uses the **Output** tab in this way.

Display 12.9 Generated Transformation in a Sample Job



5 Open the **Options** tab in the properties window for the generated transformation. Enter appropriate values for each of the options created for the transformation. The values for the sample job are shown in the following display.

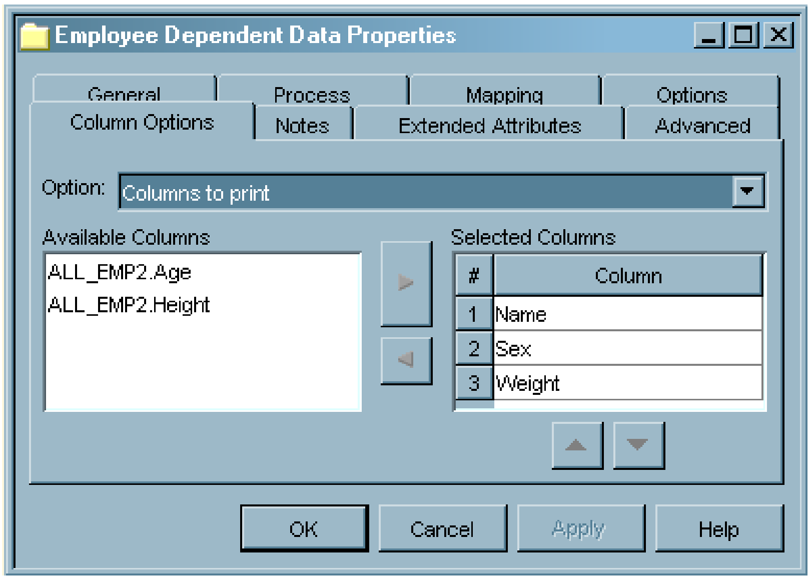
Display 12.10 Option Values in a Sample Job



Option Name	Option Value
Create SYSLAST Macro Variable	Yes
Report title	Employee Dependent Data

Click **`Column Options`** to select the columns that are displayed in the **`Output`** tab of the Process Designer window. The selections for the sample job are depicted in the following display.

Display 12.11 Column Options in a Sample Job



Option: Columns to print

Available Columns

ALL_EMP2.Age
ALL_EMP2.Height

Selected Columns

#	Column
1	Name
2	Sex
3	Weight

With these settings, the following %LET statement is generated when you run the job:

```
%let ReportTitle = %nrquote(Employee Dependent Data);
 %let ColumnsToPrint = "Name"n "Age"n "Weight"n;
```

Click **`OK`** close the properties window and save the settings.

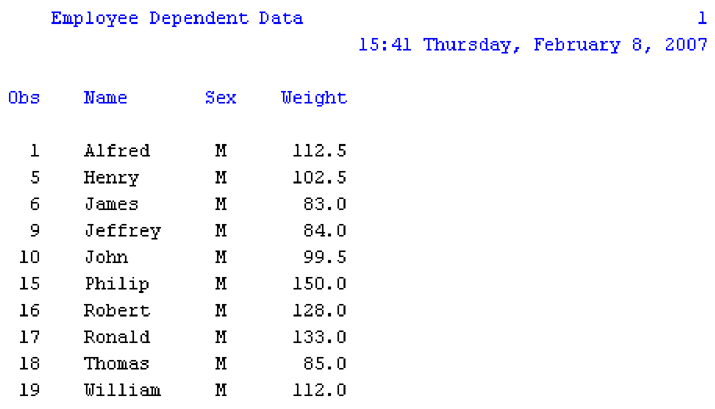
6 Run the job by right-clicking inside the Process Designer and selecting **`Submit`** from the pop-up menu. SAS Data Integration Studio generates and runs the following code:

```
/* Options  */
 %let ReportTitle = %nrquote(Employee Dependent Data);
 %let ColumnsToPrint = "Name"n "Sex"n "Weight"n;
```

```
PROC PRINT DATA=&SYSLAST
 VAR &ColumnsToPrint;
 WHERE Sex="M" and Weight > 65;
 Title "&ReportTitle";
RUN;
```

7 After the code has executed, check the Process Designer window **Output** tab for the report shown in the following display.

Display 12.12 Sample Output Report



```
          Employee Dependent Data                             1
                                   15:41 Thursday, February 8, 2007

      Obs    Name       Sex    Weight

        1    Alfred      M      112.5
        5    Henry       M      102.5
        6    James       M       83.0
        9    Jeffrey     M       84.0
       10    John        M       99.5
       15    Philip      M      150.0
       16    Robert      M      128.0
       17    Ronald      M      133.0
       18    Thomas      M       85.0
       19    William     M      112.0
```

Additional Information

For detailed information about the Transformation Generator wizard, see the "Maintaining Generated Transformations" topic in the SAS Data Integration Studio Help. For more information about setting options on the Options screen, see the "Create/Edit Option Window" Help topic.

Maintaining a Generated Transformation

Problem

You want to perform maintenance tasks such as the following:

□ Analyzing the Impact of Generated Transformations

□ Importing Generated Transformations Generated BeforeSAS Data Integration Studio 3.4

□ Updating Generated Transformations

These tasks are necessary because of (1) the way that generated transformations function in jobs and (2) the effects that changes made to the transformations could have on the jobs that contain them.

Solution

You can update a generated transformation, but the change can affect the existing jobs that include that transformation. Before you change a generated transformation, you should run impact analysis on that transformation to see all of the jobs that would be affected by the change.

After you have run impact analysis, you can make updates to the transformations. Changes to a generated transformation can affect existing jobs that include that transformation. They can also affect any new jobs that will include that transformation. Therefore, you should be very careful about any generated transformation that has been included in existing jobs. This precaution reduces the possibility that any one user could make changes to a generated transformation that would adversely affect many users.

Tasks

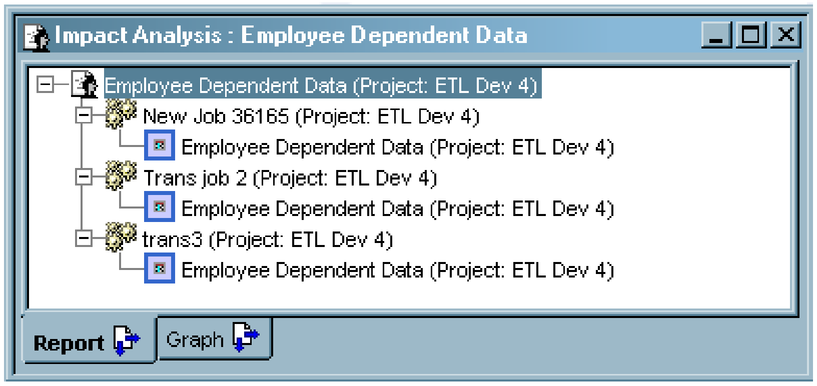
Analyze the Impact of Generated Transformations

Perform the following steps to run impact analysis on a generated transformation:

1 Find the generated transformation that you want to analyze in the Process Library tree.

2 Right-click the transformation and click **`Impact Analysis`**. (You can also click **`Impact Analysis`** in the **`Tools`** menu.) The **`Report`** tab of the Impact Analysis window displays, as shown in the following display.

Display 12.13 Impact Analysis on a Sample Generated Transformation



Impact Analysis : Employee Dependent Data

```
Employee Dependent Data (Project: ETL Dev 4)
    New Job 36165 (Project: ETL Dev 4)
        Employee Dependent Data (Project: ETL Dev 4)
    Trans job 2 (Project: ETL Dev 4)
        Employee Dependent Data (Project: ETL Dev 4)
    trans3 (Project: ETL Dev 4)
        Employee Dependent Data (Project: ETL Dev 4)
```

The selected generated transformation is named Employee Dependent Data. The Impact Analysis window shows that the selected transformation is used in the several job. You can right-click the objects in the **`Report`** tab to access their properties windows and obtain information about them. For details about the available options, see the "Report Tab options" topic in the SAS Data Integration Studio help. For a data-flow view of the impacts, select the **`Graph`** tab.

Import Generated Transformations Exported Before SAS Data Integration Studio 3.4

Perform the following steps to import a generated transformation that was exported from a version of SAS Data Integration Studio earlier than 3.4:

1 Select **Transformation Importer** from the **Tools** menu. The Transformation Importer window appears.

2 Build a list of XML files to import by performing these steps one or more times, as follows:

   □ Click **Add**. An Import Transform window appears.

   □ Browse for and select an XML file that represents a transformation, and click **OK**.

3 Click **OK** in the Transformation Importer window to import the transformations.

Update Generated Transformations

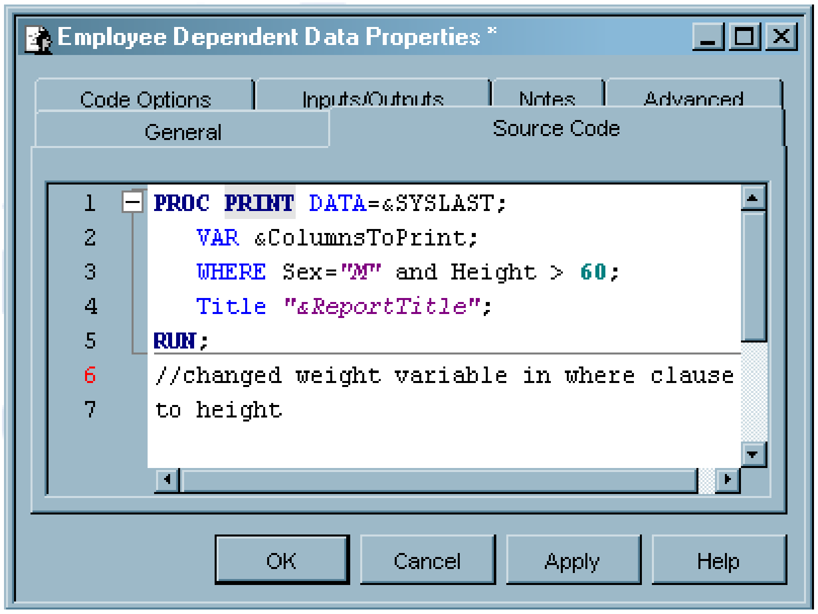
Perform the following steps to update the source code and other properties of a generated transformation.

1 Access the properties window of the transformation that you want to update by double-clicking on the transformation's name.

2 Click on a tab that you want to update.

3 Make any needed changes to the source code. Click **Apply** to save these changes to the SAS code. The following display depicts an update to the source code of a sample transformation.

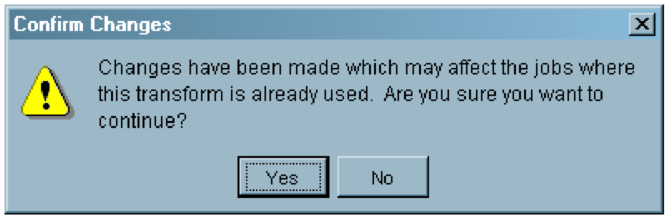
Display 12.14 Sample Source Code Tab With Updates



```
PROC PRINT DATA=&SYSLAST;
   VAR &ColumnsToPrint;
   WHERE Sex="M" and Height > 60;
   Title "&ReportTitle";
RUN;
//changed weight variable in where clause
to height
```

Note: Any change that you make to the generated transformation can affect existing jobs that contain the transformation. Therefore, the warning in the following display is shown. △

Display 12.15 Confirm Changes Warning


Confirm Changes

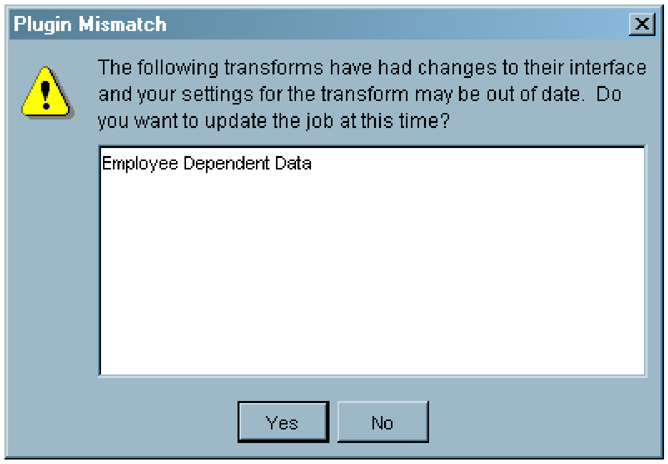
Changes have been made which may affect the jobs where this transform is already used. Are you sure you want to continue?

4 Make any updates that are needed to the other tabs in the properties window.

5 Click **OK** to save the updates and exit the transformation properties window.

Note: Another warning is shown when run a job that contains an updated generated transformation. This warning is depicted in the following display. △

Display 12.16 Plugin Mismatch Warning


Plugin Mismatch

The following transforms have had changes to their interface and your settings for the transform may be out of date. Do you want to update the job at this time?

Employee Dependent Data

CHAPTER
13

Optimizing Process Flows

About Process Flow Optimization

Efficient process flows are critical to the success of any data management project, especially as data volumes and complexity increase. This chapter begins with a description of how to improve the performance of process flows in SAS Data Integration Studio with the following techniques:

- manage process data
- manage columns
- streamline process flow components

The remainder of the chapter explains how to analyze the performance of process flows that have already been created by with the following techniques:

- use simple debugging techniques
- use SAS logs
- review temporary output tables

Managing Process Data

Problem

You want to optimize a process flow that is running too slowly or generating intermediate files that are clogging your file storage system.

Solution

You can perform the following tasks that can help manage process data effectively:

- manage temporary and permanent tables
- manage views and physical tables
- delete intermediate files
- cleanse and validate data
- minimize remote data access

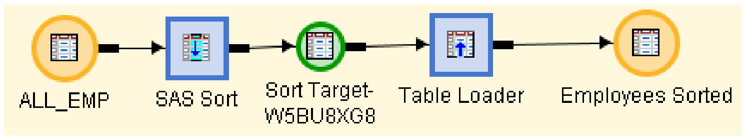
Tasks

Manage Temporary and Permanent Tables for Transformations

By default, most transformations in a SAS Data Integration Studio job write their output to a temporary location, the SAS Work library. Later in the process flow, a loading transformation, such as the Table Loader, reads the temporary output table and writes the data to a permanent data store.

In the following display, a SAS Sort transformation reads data from the ALL_EMP table, sorts it, and writes its output to a temporary table named Sort Target-W5BU8XG8. The Table Loader transformation reads the temporary output table and writes the data to a permanent data store named Employees Sorted.

Display 13.1 Transformation Sends Output to a Temporary Table



The default combination of a temporary output table and a separate loading transformation gives you the most control over how output is passed from a transformation to the next step in a process flow. Also, there are situations where a separate loading transformation is required.

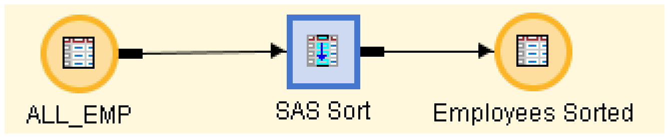
A separate loading transformation is required in order to do any of the following tasks:

- load a DBMS table
- load tables that contain rows that must be updated upon load (instead of dropping and recreating the table each time the job is executed)
- create primary keys, foreign keys, or column constraints
- perform operations on constraints before or after the loading of the output table
- perform operations on indexes other than after the loading of the output table

If you do not have to do any of these tasks, consider deleting the default temporary output table and Table Loader transformation from the process flow, as described in "Delete Intermediate Files" on page 241.

If a transformation does not need to perform any of the special load tasks that are described in the previous section, consider deleting the default temporary output table and Table Loader transformation from the process flow, as shown in the following display.

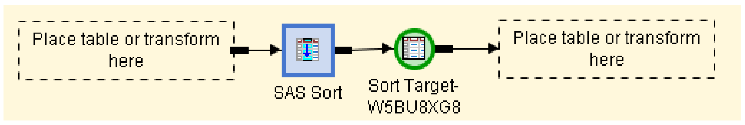
Display 13.2 Transformation Sends Output Directly to a Permanent Data Store



A process flow without temporary outputs might be more efficient than a process flow with temporary outputs. However, the simpler process flow does not give you much control over how output is passed to the next step in a process flow. Use the simpler flow when output will be loaded into a SAS table, and when it is acceptable for the output table to be dropped and recreated each time the job is executed.

There are a number of ways to create a process flow in which a transformation writes directly to a permanent data store rather than to a temporary table. For example, if you select and drag a transformation from the Process Library into the Process Designer window, the transformation will have a temporary output table, as shown in the following display.

Display 13.3 Transformation Template with a Temporary Output Table



You can replace a temporary table by dragging the icon for a permanent table and dropping it on the icon for the temporary table. For example, you could select and drag the icon for the Employees Sorted table from the Inventory tree and drop the table on the icon for the temporary output table (Sort Target-W5BU8XG8). The resulting process flow will resemble Display 13.2 on page 239.

As another example, suppose that you have the populated job shown in Display 13.1 on page 239. If you delete the Table Loader transformation, the temporary output table will be deleted as well, and SAS Sort will load directly to the Employees Sorted table.

Manage Views and Physical Tables

In general, each step in a process flow creates an output table that becomes the input for the next step in the flow. Consider what format would be best for transferring data between steps in the flow. There are two choices:

- write the output for a step to disk (in the form of SAS data files or RDBMS tables)
- create views that process input and pass the output directly to the next step, with the intent of bypassing some writes to disk

SAS supports two types of views, SQL views and DATA Step views. The two types of views can behave differently. Switching from views to physical tables or tables to views sometimes makes little difference in a process flow. At other times, improvements can be significant. The following tips are useful:

- If the data that is defined by a view is referenced only once in a process flow, then a view is usually appropriate.
- If the data that is defined by a view is referenced multiple times in a process flow, then putting the data into a physical table will likely improve overall performance. As a view, SAS must execute the underlying code repeatedly each time the view is accessed.
- If the view is referenced once in a process flow, but the reference is a resource-intensive procedure that performs multiple passes of the input, then consider using a physical table.
- If the view is SQL and is referenced once, but the reference is another SQL view, then consider using a physical table. SAS SQL optimization can be less effective when views are nested. This is especially true if the steps involve joins or RDBMS sources.
- If the view is SQL and involves a multi-way join, it is subject to performance limitations and disk space considerations.

Assess the overall impact to your process flow if you make changes based on these tips. In some circumstances, you might find that you have to sacrifice performance in order to conserve disk space.

You can right-click a temporary output table in the Process Designer window to access the `Create View` option. Then, you can select and deselect this option to switch between physical tables and views. In this way, you can test the performance of a process flow while you switch between tables and views.

Delete Intermediate Files

Transformations in a SAS Data Integration Studio job can produce the following types of intermediate files:

- procedure utility files that are created by the SORT and SUMMARY procedures when these procedures are used in the transformation
- transformation temporary files that are created by the transformation as it is working
- transformation output tables that are created by the transformation when it produces its result; the output for a transformation becomes the input to the next transformation in the flow

By default, procedure utility files, transformation temporary files, and transformation output tables are created in the WORK library. You can use the -WORK invocation option to force all intermediate files to a specified location. You can use the -UTILLOC invocation option to force only utility files to a separate location.

Knowledge of intermediate files helps you to perform the following tasks:

- view or analyze the output tables for a transformation and verify that the output is correct
- estimate the disk space that is needed for intermediate files

These intermediate files are usually deleted after they have served their purpose. However, it is possible that some intermediate files might be retained longer than desired in a particular process flow. For example, some user-written transformations might not delete the temporary files that they create.

Utility files are deleted by the SAS procedure that created them. Transformation temporary files are deleted by the transformation that created them. When a SAS Data Integration Studio job is executed in batch, transformation output tables are deleted when the process flow ends or the current server session ends.

When a job is executed interactively in SAS Data Integration Studio, transformation output tables are retained until the Process Designer window is closed or the current server session is ended in some other way (for example, by selecting **Process ▶ Kill** from the menu. Transformation output tables can be used to debug the transformations in a job, as described in "Reviewing Temporary Output Tables" on page 251. However, as long as you keep the job open in the Process Designer window, the output tables remain in the WORK library on the SAS Workspace Server that executed the job. If this is not what you want, you can manually delete the output tables, or you can close the Process Designer window and open it again, which will delete all intermediate files.

Here is a post-processing macro that can be incorporated into a process flow. It uses the DATASETS procedure to delete all data sets in the Work library, including any intermediate files that have been saved to the Work library.

```
%macro clear_work;
 %local work_members;
 proc sql noprint;
 select memname
```

```
     into :work_members separated by ","
     from dictionary.tables
     where
      libname = "WORK" and
      memtype = "DATA";
     quit;
     data _null_;
      work_members = symget("work_members");
      num_members = input(symget("sqlobs"), best.);
      do n = 1 to num_members;
       this_member = scan(work_members, n, ",");
       call symput("member"||trim(left(put(n,best.))),trim(this_member));
      end;
      call symput("num_members", trim(left(put(num_members,best.))));
     run;
     %if #_members gt 0 %then %do;
      proc datasets library = work nolist;
       %do n=1 %to #_members;
        delete &&member&n
       %end;
      quit;
     %end;
    %mend clear_work;
    %clear_work
```

Note: The previous macro deletes all data sets in the Work library. △

For details about adding a post process to a SAS Data Integration Studio job, see "Specifying Options for Jobs" on page 211.

The transformation output tables for a process flow remain until the SAS session that is associated with the flow is terminated. Analyze the process flow and determine whether there are output tables that are not being used (especially if these tables are large). If so, you can add transformations to the flow that will delete these output tables and free up valuable disk space and memory. For example, you can add a generated transformation that will delete output tables at a certain point in the flow. For details about generated transformations, see "Creating and Using a Generated Transformation" on page 226.

Cleanse and Validate Data

Clean and de-duplicate the incoming data early in the process flow so that extra data that might cause downstream errors in the flow is caught and eliminated quickly. This process can reduce the volume of data that is being sent through the process flow.

To clean the data, consider using the Sort transformation with the NODUPKEY option or the Data Validation transformation. The Data Validation transformation can perform missing-value detection and invalid-value validation in a single pass of the data. It is important to eliminate extra passes over the data, so try to code all of these validations into a single transformation. The Data Validation transformation also provides de-duplication capabilities and error-condition handling. For information, search for data validation in SAS Data Integration Studio help.

Minimize Remote Data Access

Remote data has to be copied locally because it is not accessible by the relevant components in the default SAS Application Server at the time that the code was generated. SAS uses SAS/CONNECT and the UPLOAD and DOWNLOAD procedures to move data. It can take longer to access remote data than local data, especially when you access large data sets.

For example, the following data is considered local in a SAS Data Integration Studio job:

- □ data that can be accessed as if it were on the same computers as the SAS Workspace Server components of the default SAS Application Server
- □ data that is accessed with a SAS/ACCESS engine (used by the default SAS Application Server)

The following data is considered remote in a SAS Data Integration Studio job:

- □ data that cannot be accessed as if it were on the same computers as the SAS Workspace Server
- □ data that exists in a different operating environment from the SAS Workspace Server components of the default SAS Application Server (such as mainframe data that is accessed by servers running under Microsoft Windows)

Avoid or minimize remote data access in a process flow. For information about accessing remote data, or executing a job on a remote host, administrators should see "Multi-Tier Environments" in the SAS Data Integration Studio chapter in the *SAS Intelligence Platform: Desktop Application Administration Guide*.

Managing Columns

Problem

Your process flows are running slowly, and you suspect that the columns in your source tables are either poorly managed or superfluous.

Solution

You can perform the following tasks on columns to improve the performance of process flows:

- □ dropping unneeded columns
- □ avoid adding unneeded columns
- □ aggregating columns
- □ matching column variables size to data length

Tasks

Drop Unneeded Columns

As soon as the data comes in from a source, consider dropping any columns that are not required for subsequent transformations in the flow. You can drop columns and make aggregations early in the process flow instead of later. This prevents the extraneous detail data from being carried along between all transformations in the flow. You should work to create a structure that matches the ultimate target table structure as closely as possible early in the process flow. Then, you can avoid carrying extra data along with the process flow.

To drop columns in the output table for a SAS Data Integration Studio transformation, click the **Mapping** tab and remove the extra columns from the **Target table** area on the tab. Use derived mappings to create expressions to map several columns together. You can also turn off automatic mapping for a transformation by right-clicking the transformation in the process flow and deselecting the **Automap** option in the pop-up menu. You can then build your own transformation output table columns to match your ultimate target table and map.

Avoid Adding Unneeded Columns

As data is passed from step to step in a process flow, columns could be added or modified. For example, column names, lengths, or formats might be added or changed. In SAS Data Integration Studio, these modifications to a table, which are done on a transformation's **Mapping** tab, often result in the generation of an intermediate SQL view step. In many situations, that intermediate step adds processing time. Try to avoid generating more of these steps than is necessary.

You should rework your flow so that activities such as column modifications or additions throughout many transformations in a process flow are consolidated within fewer transformations. Avoid using unnecessary aliases; if the mapping between columns is one-to-one, then keep the same column names. Avoid multiple mappings on the same column, such as converting a column from a numeric to a character value in one transformation and then converting it back from a character to a numeric value in another transformation. For aggregation steps, rename any columns within those transformations, rather than in subsequent transformations.

Aggregate Columns for Efficiency

When you add column mappings, also consider the level of detail that is being retained. Ask these questions:

- Is the data being processed at the right level of detail?
- Can the data be aggregated in some way?

Aggregations and summarizations eliminate redundant information and reduce the number of records that have to be retained, processed, and loaded into a data collection.

Match the Size of Column Variables to Data Length

Verify that the size of the column variables in the data collection is appropriate to the data length. Consider both the current and future uses of the data:

- Are the keys the right length for the current data?
- Will the keys accommodate future growth?
- Are the data sizes on other variables correct?
- Do the data sizes need to be increased or decreased?

Data volumes multiply quickly, so ensure that the variables that are being stored in the data warehouse are the right size for the data.

Streamlining Process Flow Components

Problem

You have worked hard to optimize the data and columns in your process flow, but your flow is still running too slowly.

Solution

You can try the following best practices when they are relevant to your process flows:

- □ Work from simple to complex.
- □ Use transformations for star schemas and lookups.
- □ Use surrogate keys.

Tasks

Work From Simple to Complex

When you build process flows, build complexity up rather than starting at a complex task. For example, consider building multiple individual jobs and validating each rather than building large, complex jobs. This will ensure that the simpler logic produces the expected results.

Also, consider subsetting incoming data or setting a pre-process option to limit the number of observations that are initially being processed in order to fix job errors and validate results before applying processes to large volumes of data or complex tasks. For details about limiting input to SAS Data Integration Studio jobs and transformations, see "Limit Input to a Transformation" on page 247.

Use Transformations for Star Schemas and Lookups

Consider using the Lookup transformation when you build process flows that require lookups such as fact table loads. The Lookup transformation is built using a fast in-memory lookup technique known as DATA step hashing that is available in SAS®9. The transformation allows for multi-column keys and has useful error handling techniques such as control over missing-value handling and the ability to set limits on errors.

When you are working with star schemas, consider using the SCD Type 2 transformation. This transformation efficiently handles change data detection, and has been optimized for performance. Several change detection techniques are supported: date-based, current indicator, and version number. For details about the SCD Type 2 transformation, see Chapter 19, "Working with Slowly Changing Dimensions," on page 343.

Use Surrogate Keys

Another technique to consider when you are building the data warehouse is to use incrementing integer surrogate keys as the main key technique in your data structures. Surrogate keys are values that are assigned sequentially as needed to populate a dimension. They are very useful because they can shield users from changes in the operational systems that might invalidate the data in a warehouse (and thereby require redesign and reloading). For example, using a surrogate key can avoid issues if the operational system changes its key length or type. In this case, the surrogate key remains valid. An operational key will not remain valid.

The SCD Type 2 transformation includes a surrogate key generator. You can also plug in your own methodology that matches your business environment to generate the keys and point the transformation to it. There is also a Surrogate Key Generator transformation that can be used to build incrementing integer surrogate keys.

Avoid character-based surrogate keys. In general, functions that are based on integer keys are more efficient because they avoid the need for subsetting or string partitioning that might be required for character-based keys. Numeric strings are also smaller in size than character strings, thereby reducing the storage required in the warehouse.

For details about surrogate keys and the SCD Type 2 transformation, see Chapter 19, “Working with Slowly Changing Dimensions,” on page 343.

Using Simple Debugging Techniques

Problem

Occasionally a process flow might run longer than you expect or the data that is produced might not be what you anticipate (either too many records or too few). In such cases, it is important to understand how a process flow works. Then, you can correct errors in the flow or improve its performance.

Solution

A first step in analyzing process flows is being able to access information from SAS that will explain what happened during the run. If there were errors, you need to understand what happened before the errors occurred. If you are having performance issues, then the logs will explain where you are spending your time. Finally, if you know what SAS options are set and how they are set, this can help you determine what is going on in your process flows. You can perform the following tasks:

- Monitor job status.
- Verify output.
- Limit input.
- Add debugging code.
- Set SAS invocation options.
- Set and check status codes.

Tasks

Monitor Job Status

You can use the Job Status Manager window to display the name, status, starting time, ending time, and application server used for all jobs submitted in the current session. You can also right-click on any row to clear, view, cancel, kill, or resubmit a job. From the SAS Data Integration Studio desktop, select **Tools ▶ Job Status Manager** to open the Job Status Manager window. For more information, see "Using the Job Status Manager" on page 159.

Verify Output From a Transformation

You can view the output tables for the transformations in the job. Reviewing the output tables enables you to verify that each transformation is creating the expected output. This review can be useful when a job is not producing the expected output or when you suspect that something is wrong with a particular transformation in the job. For more information, see "Browsing Table Data" on page 109.

Limit Input to a Transformation

When you are debugging and working with large data files, you might find it useful to decrease some or all of the data that is flowing into a particular step or steps. One way of doing this is to use the OBS= data set option on input tables of DATA steps and procedures.

To specify the OBS= for an entire job in SAS Data Integration Studio, add the following code to the **`Pre and Post Processing`** tab in the job's property window:

```
options obs=<number>;
```

To specify the OBS= for a transformation within a job, you can temporarily add the option to the system options field on the **`Options`** tab in the transformation's property window. Alternatively, you can edit the code that is generated for the transformation and execute the edited code. For more information about this method, see "Specifying Options for Jobs" on page 211.

Important considerations when you are using the OBS= system option include the following:

- □ All inputs into all subsequent steps are limited to the specified number, until the option is reset.
- □ Setting the number too low before a join or merge step can result in few or no matches, depending on the data.
- □ In the SAS Data Integration Studio Process Editor, this option stays in effect for all runs of the job until it is reset or the Process Designer window is closed.

The syntax for resetting the option is as follows:

```
options obs=MAX;
```

Note: Removing the OBS= line of code from the Process Editor does not reset the OBS= system option. You must reset it as shown previously, or by closing the Process Designer window. △

Add Debugging Code to a Process Flow

If you are analyzing a SAS Data Integration Studio job, and the information that is provided by logging options and status codes is not enough, consider the following methods for adding debugging code to the process flow.

Table 13.1 Methods for Adding Custom Debugging Code

Method	Documentation
Replace the generated code for a transformation with user-written code.	"Adding User-Written Source Code to an Existing Job" on page 219.
Add a User-Written Code transformation to the process flow.	"Creating a Job That Includes the User Written Code Transformation" on page 223.
Add a generated transformation to the process flow.	"Creating and Using a Generated Transformation" on page 226.
Add a return code to the process flow.	"Set and Check Status Codes" on page 248.

Custom code can direct information to the log or to alternate destinations such as external files, tables. Possible uses include tests of frequency counts, dumping out SAS macro variable settings, or listing the run time values of system options.

Set SAS Invocation Options on Jobs

When you submit a SAS Data Integration Studio job for execution, it is submitted to a SAS Workspace Server component of the relevant SAS Application Server. The relevant SAS Application Server is one of the following:

- the default server that is specified on the **SAS Server** tab in the Options window
- the SAS Application Server to which a job is deployed with the **Deploy for Scheduling** option

To set SAS invocation options for all SAS Data Integration Studio jobs that are executed by a particular SAS server, specify the options in the configuration files for the relevant SAS Workspace Servers, batch or scheduling servers, and grid servers. (You do not set these options on SAS Metadata Servers or SAS Stored Process Servers.) Examples of these options include UTILLOC, NOWORKINIT, or ETLS_DEBUG. For more information, see "Modifying Configuration Files or SAS Start Commands for Application Servers" on page 212.

To set SAS global options for a particular job, you can add these options to the **Pre and Post Process** tab in the Properties window for a job. For more information about adding code to this window, see "Specifying Options for Jobs" on page 211.

The property window for most transformations within a job has an **Options** tab with a **System Options** field. Use the **System Options** field to specify options for a particular transformation in a job's process flow. For more information, see "Specifying Options for a Transformation" on page 211.

For more information about SAS options, search for relevant phrases such as "system options" and "invoking SAS" in SAS OnlineDoc.

Set and Check Status Codes

When you execute a job in SAS Data Integration Studio, a return code for each transformation in the job is captured in a macro variable. The return code for the job is set according to the least successful transformation in the job. SAS Data Integration Studio enables you to associate a return code condition, such as **Successful**, with an action, such as **Send Email** or **Abort**. In this way, users can specify how a return code is handled for the job or transformation.

For example, you could specify that a transformation in a process flow will terminate based on conditions that you define. This can reduce the log to just the transformations leading up to the problem being investigated, making the log more manageable and eliminating inconsequential error messages. For more information about status code handling, see “Managing Status Handling” on page 161.

Using SAS Logs

Problem

The errors, warnings, and notes in the SAS log provide information about process flows. However, large SAS logs can decrease performance, so the costs and benefits of large SAS logs should be evaluated. For example, in a production environment, you might not want to create large SAS logs by default.

Solution

You can use SAS logs in the following ways:

- □ Evaluate SAS logs.
- □ Capture additional SAS options In the SAS log.
- □ View or hide SAS logs.
- □ Redirect large SAS logs to a file.

Tasks

Evaluate SAS Logs

The SAS logs from your process flows are an excellent resource to help you understand what is happening as the flows execute. For example, when you look at the run times in the log, compare the real-time values to the CPU time (user CPU plus system CPU). For read operations, the real time and CPU time should be close. For write operations, however, the real time can substantially exceed the CPU time, especially in environments that are optimized for read operations. If the real time and the CPU time are not close, and they should be close in your environment, investigate what is causing the difference.

If you suspect that you have a hardware issue, see *A Practical Approach to Solving Performance Problems with the SAS System*, a document that is available from the Scalability and Performance Papers page at **`http://support.sas.com/rnd/scalability/papers/`**.

If you determine that your hardware is properly configured, then review the SAS code. Transformations generate SAS code. Understanding what this code is doing is very important to ensure that you do not duplicate tasks, especially SORTs, which are resource-intensive. The goal is to configure the hardware so that there are no bottlenecks, and to avoid needless I/O in the process flows.

Capture Additional SAS Options In the SAS Log

To analyze performance, we recommend that you turn on the following SAS options so that detailed information about what the SAS tasks are doing is captured in the SAS log:

```
FULLSTIMER
MSGLEVEL=I (this option prints additional notes pertaining to index, merge
   processing, sort utilities, and CEDA usage, along with the standard notes,
   warnings, and error messages)
SOURCE, SOURCE2
MPRINT
NOTES
```

To interpret the output from the FULLSTIMER option, see *A Practical Approach to Solving Performance Problems with the SAS System*, a document that is available from the Scalability and Performance Papers page at **http://support.sas.com/rnd/scalability/papers/**.

In addition, the following SAS statements will also send useful information to the SAS log:

```
PROC OPTIONS OPTION=UTILLOC; run;
PROC OPTIONS GROUP=MEMORY; run;
PROC OPTIONS GROUP=PERFORMANCE; run;
LIBNAME _ALL_ LIST;
```

The PROC OPTIONS statement will send SAS options and their current settings to the SAS log. There are hundreds of SAS options, so if, for example, you prefer to see which value has been set to the SAS MEMORY option, you can issue the PROC OPTIONS statement with the GROUP=MEMORY parameter. The same is true if you want to see only the SAS options that pertain to performance.

The LIBNAME _ALL_ LIST statement will send information (such as physical path location and the engine being used) to the SAS log about each libref that is currently assigned to the SAS session. This data is helpful for understanding where all the work occurs during the process flow. For details about setting SAS invocation options for SAS Data Integration Studio, see “Set SAS Invocation Options on Jobs” on page 248.

View or Hide SAS Logs

The Process Designer window in SAS Data Integration Studio has a **Log** tab that displays the SAS log for the job in the window. Perform the following steps to display or hide the **Log** tab:

1 Select **Tools ▶ Options** on the SAS Data Integration Studio menu bar to display the Options window.

2 Click the **General** tab in the Options window. Then, select or deselect the check box that controls whether the **Log** tab is displayed in the Process Designer window.

3 Click **OK** in the Options window to save the setting and close the window.

Redirect Large SAS Logs to a File

The SAS log for a job provides critical information about what happened when a job was executed. However, large jobs can create large logs, which can slow down SAS Data Integration Studio. In order to avoid this problem, you can redirect the SAS log to a permanent file. Then, you can turn off the **Log** tab in the Process Designer window.

When you install SAS Data Integration Studio, the Configuration Wizard enables you to set up as permanent SAS log files for each job that is executed. The SAS log filenames will contain the name of the job that created the log, plus a timestamp of when the job was executed.

Alternatively, you can add the following code to the **Pre and Post Process** tab in the properties window for a job:

```
proc printto log=...path_to_log_file NEW; run;
```

For details about adding pre-process code to a SAS Data Integration Studio job, see "Specifying Options for Jobs" on page 211. This code will cause the log to be redirected to the specified file. Be sure to use the appropriate host-specific syntax of the host where your job will be running when you specify this log file, and make sure that you have Write access to the location where the log will be written.

Reviewing Temporary Output Tables

Problem

Most transformations in a SAS Data Integration Studio job create at least one output table. Then, they store these tables in the Work library on the SAS Workspace Server that executes the job. The output table for each transformation becomes the input to the next transformation in the process flow. All output tables are deleted when the job is finished or the current server session ends.

Sometimes a job does not produce the expected output. Other times, something can be wrong with a particular transformation. In either case, you can view the output tables for the transformations in the job to verify that each transformation is creating the expected output. Output tables can also be preserved to determine how much disk space they require. You can even use them to restart a process flow after it has failed at a particular step (or in a specific transformation).

Solution

You can (1) view a transformation's temporary output table from the Process Designer window and (2) preserve temporary output tables so that you can view their contents by other means. You can perform the following tasks to accomplish these objectives:

- Preserve temporary output tables.
- View temporary output tables.
- Redirect temporary output tables.
- Add a List Data transformation to a process flow.
- Add a User-Written Code transformation to a process flow.

Tasks

Preserve Temporary Output Tables

When SAS Data Integration Studio jobs are executed in batch mode, a number of SAS options can be used to preserve intermediate files in the Work library. These system options can be set as described in "Set SAS Invocation Options on Jobs" on page 248.

Use the NOWORKINIT system option to prevent SAS from erasing existing Work files on invocation. Use the NOWORKTERM system option to prevent SAS from erasing existing Work files on termination.

For example, to create a permanent SAS Work library in UNIX and PC environments, you can start the SAS Workspace Server with the WORK option to redirect the Work files to a permanent Work library. The NOWORKINIT and NOWORKTERM options must be included, as follows:

```
C:\>"C:\Program Files\SAS\SAS 9.1\sas.exe"
-work "C:\Documents and Settings\sasapb\My Documents\My SAS Files\My SAS Work Folder"
-noworkinit
-noworkterm
```

This redirects the generated Work files in the folder My SAS Work Folder.

To create a permanent SAS Work library in the z/OS environment, edit your JCL statements and change the WORK DD statement to a permanent MVS data set. For example:

```
 //STEP1 EXEC SDSSAS9,REGION=50M
//* changing work lib definition to a permanent data set
//SDSSAS9.WORK DD DSN=userid.somethin.sasdata,DISP=OLD
//* other file defs
//INFILE DD ... .
```

CAUTION:
If you redirect Work files to a permanent library, you must manually delete these files to avoid running out of disk space. △

View Temporary Output Tables

Perform the following steps to view the output file:

1. Open the job in the Process Designer window.
2. Submit the job for execution. The transformations must execute successfully. (Otherwise, a current output table is not available for viewing.)
3. Right-click the transformation of the output table that you want to view, and select View Data from the pop-up menu. The transformation's output table is displayed in the View Data window.

This approach works if you do not close the Process Designer window. When you close the Process Designer window, the current server session ends, and the output tables are deleted. For information, see "Browsing Table Data" on page 109.

Redirect Temporary Output Tables

The default name for a transformation's output table is a two-level name that specifies the Work libref and a generated member name, such as work.W54KFYQY. You can specify the name and location of the output table for that transformation on the **Physical Storage** tab on the properties window of the temporary output table. Note that this location can be a SAS library or RDBMS library. This has the added benefit of providing users the ability to specify which output tables they want to retain and to allow the rest to be deleted by default. Users can use this scheme as a methodology for checkpoints by writing specific output tables to disk when needed.

Note: If you want to save a transformation output table to a library other than the SAS User library, replace the default name for the output table with a two-level name. △

If you refer to an output table with a single-level name (for example, employee), instead of a two-level name (for example, work.employee), SAS automatically sends the output table into the User library, which defaults to the Work library. However, this default behavior can be changed by any SAS user. Through the USER= system option, a SAS user can redirect the User library to a different library. If the USER= system option is set, single-level tables are stored in the User library, which has been redirected to a different library, instead of to the Work library.

Add the List Data Transformation to a Process Flow

In SAS Data Integration Studio, you can use the List Data transformation to print the contents of an output table from the previous transformation in a process flow. Add the List Data transformation after any transformation whose output table is of interest to you.

The List Data transformation uses the PRINT procedure to produce output. Any options associated with that procedure can be added from the **`Options`** tab in the transformation's property window. By default, output goes to the **`Output`** tab in the Process Designer window. Output can also be directed to an HTML file. For large data, customize this transformation to print just a subset of the data. For details, see the "Example: Create Reports from Table Data" topic in SAS Data Integration Studio help.

Add a User-Written Code Transformation to the Process Flow

You can add a User Written Code transformation to the end of a process flow that moves or copies some of the data sets in the Work library to a permanent library. For example, assume that there are three tables in the Work library (test1, test2, and test3). The following code moves all three tables from the Work library to a permanent library named PERMLIB and then deletes them from the Work library:

```
libname permlib base
"C:\Documents and Settings\ramich\My Documents\My SAS Files\9.1";
proc copy move
in = work
out = permlib;
select test1 test2 test3;
run;
```

For information about User Written Code transformations, see "Creating a Job That Includes the User Written Code Transformation" on page 223.

Additional Information

The techniques covered in this chapter address general performance issues that commonly arise for process flows in SAS Data Integration Studio jobs. For specific information about the performance of the SQL Join transformation, see "Optimizing SQL Processing Performance" on page 320. For specific information about the performance of the Table Loader transformation, see "Selecting a Load Technique" on page 270 and "Removing Non-Essential Indexes and Constraints During a Load" on page 272.

You can also access a library of SAS Technical Papers that cover a variety of performance-related topics. You can find these papers at **`http://support.sas.com/documentation/whitepaper/technical/`**.

CHAPTER 14
Using Impact Analysis

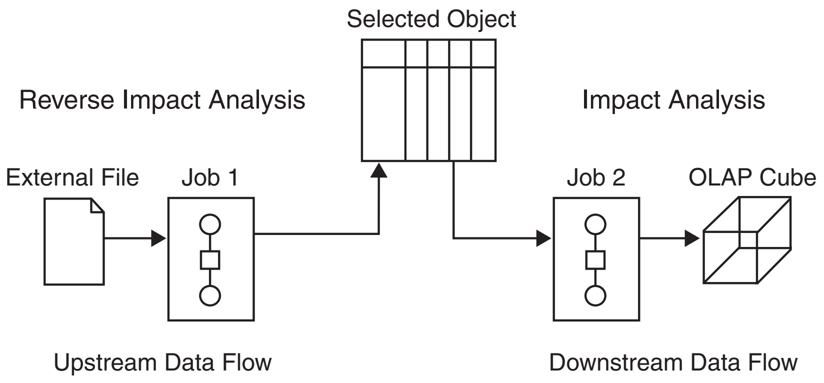
About Impact Analysis and Reverse Impact Analysis

Impact analysis identifies the tables, columns, jobs, and transformations that are *affected by* a change in a selected table or column. Use impact analysis before changing or deleting a metadata object to see how that change can affect other objects.

Reverse impact analysis identifies the tables, columns, jobs, and transformations that *contribute to* the content of a selected table or column. Use reverse impact analysis to help determine the cause of data anomalies.

The following figure shows the difference between impact analysis and reverse impact analysis for a selected object.

Figure 14.1 Differentiating Impact Analysis and Reverse Impact Analysis



As shown in the figure, impact analysis traces the impact of the selected object on later objects in the data flow. Reverse impact analysis traces the impact that previous objects in the data flow have had on the selected object.

In SAS Data Integration Studio, you can perform impact analysis and reverse impact analysis on the following kinds of metadata:

- table metadata
- column metadata
- external file metadata
- OLAP cube metadata
- metadata for OLAP cube features, levels, and measures
- metadata for Information Maps, Enterprise Guide projects, or Add-in for Microsoft Office (AMO) objects
- message queue metadata
- metadata for SAS Data Integration Studio jobs
- metadata for transformations, including generated transformations.

For further information, see the SAS Data Integration Studio Help topic on “Using Impact Analysis and Reverse Impact Analysis.”

Prerequisites

To ensure that your analyses include a search that includes all repositories, select **Tools ▶ Options** and click the `Impact Analysis` tab. In the `Impact Analysis` tab, select the check box `Continue analysis into dependent repositories`.

Impact analysis and reverse impact analysis display only the objects for which users have ReadMetadata permission. Administrators should make sure that users have the ReadMetadata permissions for all repositories needed for their analysis. For more information, the administrator should see *SAS Intelligence Platform: Security Administration Guide*.

Performing Impact Analysis

Problem

A table is used in the process flow for a job. You want to delete the metadata for a column in a table, and you want to trace the impact this will have on later objects in the process flow.

Solution

Use impact analysis to trace the impact of the selected object on later objects in the process flow for the job.

Tasks

Perform Impact Analysis

In general, to perform impact analysis on a metadata object, right-click the object in a tree view or in the context of a process flow in the Process Designer window, then select **Impact Analysis** from the pop-up menu.

Alternatively, you can select the object in a tree view or in the context of a process flow, then select **Tools ▶ Impact Analysis** from the menu bar.

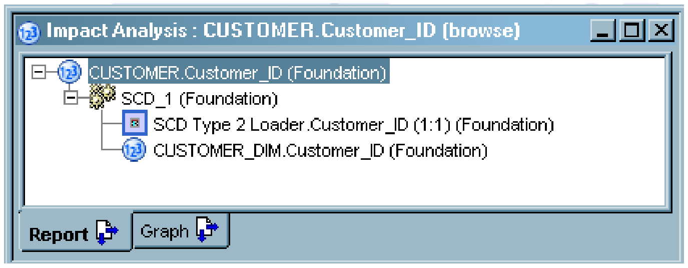
Perform the following steps to trace the impact of the metadata for a table column:

1 Double-click the metadata object for the table that contains the column to be analyzed. The properties window for the table displays.

2 Select the **Columns** tab.

3 On the **Columns** tab, right-click the column to be analyzed and select **Impact Analysis** from the pop-up menu. In the resulting Impact Analysis window, the **Report** tab displays by default. In the following display, the **Report** tab shows the result of an analysis performed on a column named Customer_ID in a table named CUSTOMER.

Display 14.1 Report Tab in the Impact Analysis Window


Impact Analysis : CUSTOMER.Customer_ID (browse)

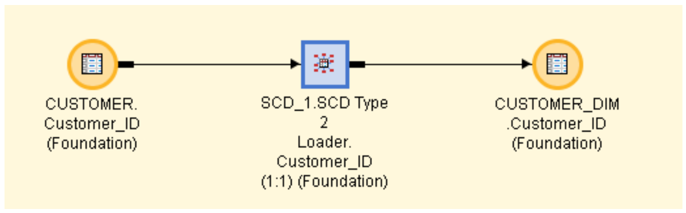
```
CUSTOMER.Customer_ID (Foundation)
  SCD_1 (Foundation)
    SCD Type 2 Loader.Customer_ID (1:1) (Foundation)
    CUSTOMER_DIM.Customer_ID (Foundation)
```

The **`Report`** tab uses a hierarchical list to illustrate the impact of the selected object (Customer_ID column) on later objects in a process flow. In the previous display, the tab contains the following objects:

- **`CUSTOMER.Customer_ID (Foundation)`**: specifies the selected column, Customer_ID, in the table CUSTOMER, which is registered in the Foundation repository.
- **`SCD_I (Foundation)`**: specifies the job, SCD_1, to which the Customer_ID column is an input.
- **`SCD Type 2 Loader.Customer_ID (1:1) (Foundation)`**: specifies the transformation which maps data from the Customer_ID column to a column later in the process flow. The mapping type is 1:1.
- **`CUSTOMER_DIM.Customer_ID (Foundation)`**: specifies the target column, Customer_ID, in the table CUSTOMER_DIM. The target column is loaded with data from the selected column.

4 Click the **`Graph`** tab to display a graphical view of the analytical results, as shown in the following example.

Display 14.2 Graph Tab of the Impact Analysis window



The **`Graph`** tab uses a process flow to illustrate the impact of the selected object (Customer_ID column) on later objects in the flow.

Performing Impact Analysis on a Generated Transformation

Problem

You want to determine how many jobs are impacted by a change to a generated transformation.

A generated transformation is a transformation that you create with the Transformation Generator wizard. You can use this wizard to create your own generated transformations and register them in your metadata repository. After they are registered, your transformations display in the Process Library, where they are available for use in any job. For more information about these transformations, see "Creating and Using a Generated Transformation" on page 226.

When you change or update a generated transformation, the change can affect the jobs that include that transformation. Before you change a generated transformation, you should run impact analysis on that transformation to see all of the jobs that would be affected by the change.

Solution

Run impact analysis on a generated transformation.

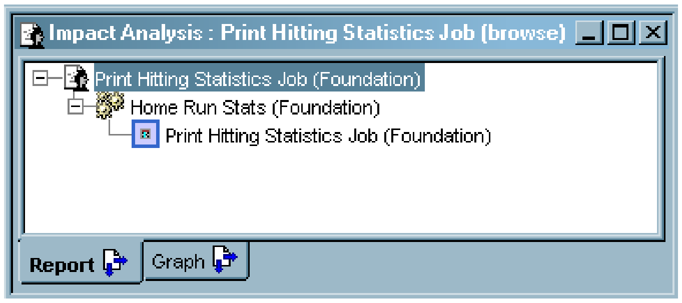
Tasks

Perform Impact Analysis on a Generated Transformation

Perform the following steps to run an impact analysis on a generated transformation:

1. From the SAS Data Integration Studio desktop, select the Process Library tree.
2. Open the folder that contains the generated transformation that you want to analyze.
3. Select that transformation and then, from the menu bar, select **Tools ▶ Impact Analysis**. The **Report** tab of the Impact Analysis window displays, as shown in the following example.

Display 14.3 Impact Analysis on a Generated Transformation


Impact Analysis : Print Hitting Statistics Job (browse)
Print Hitting Statistics Job (Foundation)
Home Run Stats (Foundation)
Print Hitting Statistics Job (Foundation)
Report
Graph

In the preceding display, the selected generated transformation is named Summary Statistics. The Impact Analysis window shows that the selected transformation is used in the job Home Run Stats.

You can right-click the objects in the **Report** tab to obtain information about those objects. For details about the available options, see the Data Integration Studio Help topic on Report Tab options.

For a process flow view of the impacts, select the **Graph** tab.

Performing Reverse Impact Analysis

Problem

A table is used in the process flow for a job. You notice an error in the data for one column, and you want to trace the data flow to that column.

Solution

Use reverse impact analysis to identify the tables, columns, jobs, and transformations that contribute to the content of a selected column.

Tasks

Perform Reverse Impact Analysis

In general, to perform impact analysis on a metadata object, right-click the object in a tree view or in the context of a process flow in the Process Designer window, then select **`Reverse Impact Analysis`** from the pop-up menu .

Alternatively, you can select the object in a tree view or in the context of a process flow, then select **Tools ▶ Reverse Impact Analysis** from the menu bar.

The steps for performing reverse impact analysis on a column are similar to the steps in “Perform Impact Analysis” on page 257.

P A R T 3

Working with Specific Transformations

CHAPTER 15 Working with Loader Transformations

About Loader Transformations

SAS Data Integration Studio provides the following transformations for loading output tables in a process flow:

- the SCD Type 2 Loader transformation, which loads source data into a dimension table, detects changes between source and target rows, updates change tracking columns, and applies generated key values. This transformation implements slowly changing dimensions. For more information see Chapter 19, "Working with Slowly Changing Dimensions," on page 343.
- the SPD Server Table Loader transformation, which reads a source and writes to an SPD Server target. This transformation is automatically added to a process flow when an SPD Server table is specified as a source or as a target. It enables you to specify options that are specific to SPD Server tables.
- the Table Loader transformation, which reads a source table and writes to a target table. This transformation is added automatically to a process flow when a table is dropped on to a Process Designer window.

About the SPD Server Table Loader Transformation

The SPD Server Table Loader transformation is automatically added to a process flow when a SAS Scalable Performance Data (SPD) Server table is added as a source or as a target in the Process Designer window. The SPD Server Table Loader generates code that is appropriate for the special data format that the server uses. It also enables you to specify options that are unique to SPD Server tables.

You can specify a variety of table options in the **`Additional Data Table Options`** field on the **`Options`** tab of the SPD Server Table Loader properties window. These options are described in detail in the documentation that is installed with the SPD Server. One example of an additional table option is the MINMAXVARLIST option that is described in the SAS Data Integration Studio Usage Notes topic in SAS Data Integration Studio help.

About the Table Loader Transformation

The Table Loader transformation provides load options and combinations of load options. You can use it in the following ways:

- You can update-and-insert rows, and you can also update only or insert only by using skip options.
- After you find and match new records, you can process them by using SQL Insert or the APPEND procedure.
- SQL Update has *ignore blanks* options similar to those for DATA Step Modify in previous releases.
- For SAS tables and for some database tables, indexes and constraints can be revised without having to recreate a table.
- For SAS tables and for some database tables, indexes and constraints can be dropped or added before or after a load. For example, Job One can load Table A without an index, and Job Two can update the same table and add the indexes.
- Index dropping is available for all update methods, not just the APPEND procedure. If an index is needed for the update method, that index will not be dropped.

Finally, the current version of the Table Loader transformation inserts a comment into the generated code for queries that states that the code was generated by version 2 of the transformation. This comment distinguishes the code from the code that is generated by the earlier version of the Table Loader transformation.

To understand how the Table Loader transformation functions in the context of a job, you need to understand the following points about how jobs are constructed:

- When a transformation is added to a process flow, it sends its output to a temporary output table by default. Each temporary output table has its own icon in the process flow. Having a separate metadata object for each temporary table makes it easier to manage the data and metadata for these tables.
- When you drop a permanent data structure such as a physical table on to the output drop zone for the transformation, the Table Loader transformation is added to the process flow. It is located between the temporary output table and the permanent data structure.

When a process flow contains a temporary output table and a Table Loader transformation, you might often need to verify or create the column mappings between the temporary output table and the permanent data structure (output table) attached to the Table Loader. Performance could be degraded for the job because of the overhead associated with the temporary output table and the Table Loader transformation. Therefore, you have the option of deleting the Table Loader transformation. For information about deleting the Table Loader transformation, "Manage Temporary and Permanent Tables for Transformations" on page 239.

Setting Table Loader Transformation Options

Problem

You want to specify options that control how the Table Loader transformation updates the target.

Solution

You can use the settings available on the **`Load Technique`** tab in the properties window for the Table Loader transformation. Some of the settings available on the tab vary depending on which load styles that you use, although some settings appear for more than one load style. Some options are set in the Advanced Load Options window.

Tasks

Set Options Based on Load Style

When you use the append to existing table load style, you can set the options listed in following table.

Table 15.1 Append to Existing Table Load Style Options

Setting	Location (group box or field)	Description
Append to Existing	Load Style	Creates the output table if it does not exist; then appends input data rows to the output table.
Append (Proc Append)	New Rows	Appends rows by using PROC APPEND. This is the default.
Insert SQL	New Rows	Appends rows by using SQL Insert code.

When you use the replace table load style, you can set the options listed in following table.

Table 15.2 Replace Table Load Style Options

Setting	Location (group box or field)	Description
Replace	Load Style	Specifies that the output table or rows in the output table are replaced.
Entire table	Replace	Replaces the entire output table by deleting and recreating the table before adding new rows. This is the default for SAS data sets.
All rows using delete	Replace	Replaces all rows in the output table by first deleting all rows before adding new rows. This is the default for DBMS tables that do not support truncation.
All rows using truncate	Replace	Replaces all rows in the output table by using the truncate method of the DBMS before adding new rows. This is the default for DBMS tables that support truncation.
Append (Proc Append)	New rows	Inserts new rows into the empty table by using PROC APPEND. This is the default.
Insert (SQL)	New rows	Inserts new rows into the empty table by using SQL Insert code.

When you use the update/insert load style, you can set the options listed in the following table.

Table 15.3 Update/Insert Load Style

Setting	Location (group box or field)	Description
Update/Insert	Load Style	Specifies that existing rows are updated, new rows are inserted, or both.
SQL Set	Matching rows	Specifies that PROC SQL is used to update output table rows with all matching input data. Also enables the **`Match by Columns`** group box, which allows you to select columns.
Modify by Column(s)	Matching rows	Specifies that the DATA STEP MODIFY BY method is used to update output table rows with all matching input data. Also enables the **`Match by Columns`** group box, which allows you to select columns.

Setting	**Location (group box or field)**	**Description**
Modify Using Index	Matching rows	Specifies that the DATA STEP MODIFY with KEY= method will be used to update the output table rows with all matching input data. Also enables the **`Modify Using Index`** group box, the **`Index`** field, and the **`Return to the top of the index for duplicate values coming from the input data`** check box.
Index	Modify Using Index Group box	Enables you to select an index by using a drop-down menu.
Return to the top of the index for duplicate values coming from the input data	Modify Using Index Group box	When selected, processes duplicate values coming from the input data by returning the update or insert operation to the top of the index.
Skip Matching Rows	Matching rows	Ignores any input rows that match rows in the output table (by selected match-columns or index). This prevents any existing rows in the output table from being modified.
Blanks can replace non-blank values	Technique(s) group box	When selected, specifies that blanks can replace non-blank values during updates and insertions.
Append (Proc Append)	New rows	Appends any unmatched rows to the output table. This is the default.
Insert (SQL)	New rows	Uses SQL to insert unmatched rows into the output table. Available only when you select **`SQL Set`** or **`Skip Matching Rows`** in the **`Matching rows`** field.
Modify by Column(s)	New rows	Adds any unmatched rows into the output table. Available only when you select **`Modify by Column(s`**) or **`Skip Matching Rows`** in the **`Matching rows`** field.
Modify Using Index	New rows	Adds any unmatched rows into the output table. Available only when you select **`Modify Using Index`** or **`Skip Matching Rows`** in the **`Matching rows`** field.
Skip New Rows	New rows	Ignores any input rows that do not match rows in the output table (by selected match-columns or index).

Setting	Location (group box or field)	Description
Available columns	Match by Column(s)	Lists the columns that are available for matching.
Column(s) to match	Match by Column(s)	Specifies the columns to match during updates and insertions to the output table.

Set Constraint Condition and Index Condition Options

The settings in the **`Constraint Condition`** and **`Index Condition`** group boxes display only when the output table for the Table Loader transformation contains one or more constraints or indexes. The options available in the **`Constraint Condition`** group box are listed in following table.

Table 15.4 Constraint Condition Options

Setting	Location (group box or field)	Description
On table creation	Before load	Creates constraints on the output table before the load but only during the run when the table is first created. This is the default.
As is (do nothing) or Leave Off	Before load	Does not generate SAS code related to adding or deleting constraints before the load. Alternatively worded as **`Leave Off`** if the selected load technique is **`Replace entire table`**.
Off	Before load	Identifies and removes constraints from the pre-existing output table before the load. Not applicable when the load technique is **`Replace entire table`**.
On refresh	Before load	Deletes obsolete constraints and recreates the constraints based on the current key definitions before the load. Not applicable when the load technique is **`Replace entire table`**.
As is (do nothing), Leave on, or Leave off	After Load	Does not generate SAS code related to adding or deleting constraints before the load. Alternatively worded as **`Leave on`** if the before action adds constraints and as **`Leave off`** if the before action removes constraints. This is the default.
Off	After Load	Identifies and removes constraints after the load.

Setting	Location (group box or field)	Description
On table creation	After load	Creates constraints on the output table after the load but only during the run when the table is first created.
On refresh or On	After load	**`On/refresh`** deletes obsolete constraints and recreates the constraints based on the current key definitions. Available as **`On/Refresh`** if the before action is **`As-is`**, and as **`On`** if the before action is **`Off`**. Not applicable when the load technique is **`Replace entire table`**.

The options available in the **`Index Condition`** group box are listed in following table.

Table 15.5 Index Condition Options

Setting	Location	Description
On table creation	Before load	Creates indexes on the output table before the load but only during the run when the table is first created. This is the default.
As is (do nothing) or Leave Off	Before load	Does not generate SAS code related to adding or deleting indexes before the load. Alternatively worded as **`Leave Off`** if the selected load technique is **`Replace entire table`**.
Off	Before load	Identifies and removes indexes from the pre-existing output table before the load. Not applicable when the load technique is **`Replace entire table`**.
On refresh	Before load	Deletes obsolete indexes and recreates the indexes based on the current key definitions before the load. Not applicable when the load technique is **`Replace entire table`**.
As is (do nothing), Leave on, or Leave off	After Load	Does not generate SAS code related to adding or deleting indexes before the load. Alternatively worded as **`Leave On`** if the before action adds constraints and as **`Leave Off`** if the before action removes constraints. This is the default.
Off	After Load	Identifies and removes indexes after the load.

Setting	Location	Description
On table creation	After load	Creates indexes on the output table after the load but only during the run when the table is first created.
On refresh or On	After load	**On/refresh** deletes obsolete indexes and recreates the indexes based on the current key definitions. Available as **On/Refresh** if the before action is **As-is**, and as **On** if the before action is **Off**. Not applicable when the load technique is **Replace entire table**.

Selecting a Load Technique

Problem

You want to load data into a permanent physical table that is structured to match your data model. As the designer or builder of a process flow in SAS Data Integration Studio, you must identify the load style that your process requires. Then you can proceed to (1) append all of the source data to any previously loaded data, (2) replace all previously loaded data with the source data, or (3) use the source data to update and add to the previously loaded data based on specific key columns. When you know the load style that is required, you can select the techniques and options that will maximize the step's performance.

Solution

You can use the Table Loader transformation to perform any of the three load styles. The transformation generates the code that is required in order to load SAS data sets, database tables, and other types of tables, such as an Excel spreadsheet. When you load tables, you can use the transformation to manage indexes and constraints on the table that is being loaded.

You select the load style in the **Load style** field on the **Load Technique** tab of the Table Loader transformation. After you have selected the load style, you can choose from a number of load techniques and options. Based on the load style that you select and the type of table that is being loaded, the choice of techniques and options will vary. The Table Loader transformation generates code to perform a combination of the following loading tasks:

- Remove all rows.
- Add new rows.
- Match and update rows.

The following sections describe the SAS code alternatives for each load task and provide tips for selecting the load technique (or techniques) that will perform best.

Tasks

Remove All Rows

This task is associated with the Replace Load style. Based on the type of target table that is being loaded, two or three of the following selections are listed in the **`Replace`** field:

- **`Replace entire table`**: Uses PROC DATASETS to delete the target table
- **`Replace all rows using truncate`**: Uses PROC SQL with TRUNCATE to remove all rows (only available for some databases)
- **`Replace all rows using delete`**: Uses PROC SQL with DELETE * to remove all rows

When you select **`Replace entire table`**, the table is removed and disk space is freed. Then the table is recreated with **`0`** rows. Consider this option unless your security requirements restrict table deletion permissions (a restriction that is commonly imposed by a database administrator on database tables). Also, avoid this method if the table has any indexes or constraints that SAS Data Integration Studio cannot recreate from metadata (for example, check constraints).

If available, consider using **`Replace all rows using truncate`**. Either of the remove all rows selections enable you to keep all indexes and constraints intact during the load. By design, using TRUNCATE is the quickest way to remove all rows. The DELETE * syntax also removes all rows; however, based on the database and table settings, this choice can incur overhead that will degrade performance. Consult your database administrator or the database documentation for a comparison of the two techniques.

CAUTION:

When DELETE * is used repeatedly to clear a SAS table, the size of that table should be monitored over time. DELETE * performs only logical deletes for SAS tables. Therefore, a table's physical size will grow, and the increased size can negatively affect performance. △

Add New Rows

For this task, the Table Loader transformation provides two techniques for all three load styles: PROC APPEND with the FORCE option and PROC SQL with the INSERT statement. The two techniques handle discrepancies between source and target table structures differently.

PROC APPEND with the FORCE option is the default. If the source is a large table and the target is in a database that supports bulk-load, PROC APPEND can take advantage of the bulk-load feature. Consider bulk-loading the data into database tables with the optimized SAS/ACCESS engine bulk loaders. (We recommended that you use native SAS/ACCESS engine libraries instead of ODBC libraries or OLEDB libraries for relational database data. SAS/ACCESS engines have native access to the databases and have superior bulk-loading capabilities.)

PROC SQL with the INSERT statement performs well when the source table is small because you don't incur the overhead needed to set up bulk-loading. PROC SQL with INSERT adds one row at a time to the database.

Match and Update Rows

The Table Loader transformation provides three techniques for matching and updating rows in a table. All the following techniques are associated with the **`Update/Insert`** load style:

- □ DATA step with the MODIFY BY option
- □ DATA step with the MODIFY KEY= option
- □ PROC SQL with the WHERE and SET statements

For each of these techniques, you must select one or more columns or an index for matching. All three techniques update matching rows in the target table. The MODIFY BY and MODIFY KEY= options have the added benefit of being able to take unmatched records and add them to the target table during the same pass through the source table.

Of these three choices, the DATA step with MODIFY KEY= option often outperforms the other update methods in tests conducted on loading SAS tables. An index is required. The MODIFY KEY= option can also perform adequately for database tables when indexes are used.

When the Table Loader uses PROC SQL with WHERE and SET statements to match and update rows, performance varies. When used in PROC SQL, neither of these statements requires data to be indexed or sorted, but indexing on the key columns can greatly improve performance. Both of these statements use WHERE processing to match each row of the source table with a row in the target table.

The update technique that you choose depends on the percentage of rows being updated. If the majority of target records are being updated, the DATA step with MERGE (or UPDATE) might perform better than the DATA step with MODIFY BY or MODIFY KEY= or PROC SQL because MERGE makes full use of record buffers. Performance results can be hardware and operating environment dependent, so you should consider testing more than one technique.

Note: The general Table Loader transformation does not offer the DATA step with MERGE as a load technique. However, you can revise the code for the MODIFY BY technique to do a merge and save that as user-written code for the transformation △

Removing Non-Essential Indexes and Constraints During a Load

Problem

You want to improve the performance of a job that includes a table that contains one or more non-essential indexes.

Solution

You can remove non-essential indexes before a load and recreate those indexes after the load. In some situations, this procedure improves performance. As a general rule, consider removing and recreating indexes if more than 10 percent of the data in the table will be reloaded.

You might also want to temporarily remove key constraints in order to improve performance. If there are significant numbers of transactions and the data that is being loaded conforms to the constraints, then removing the constraints from the target before a load removes the overhead that is associated with maintaining those constraints during the load.

To control the timing of index and constraint removal, use the options that are available on the **Load Technique** tab of the Table Loader transformation. The following settings are provided to enable you to specify the desired conditions for the constraints and indexes before and after the load:

- the **Before Load** field in the **Constraint Condition** group box
- the **After Load** field in the **Constraint Condition** group box
- the **Before Load** field in the **Index Condition** group box
- the **After Load** field in the **Index Condition** group box

The options that are available depend on the load technique that you choose. The choices translate to three different tasks: put on, take off, and leave as is. When you select **Off** for the **Before Load** options, the generated code checks for and removes any indexes (or constraints) that are found. Then, it loads the table. If an index is required for an update, that index is not removed or it will be added, as needed. Select **On** for the **After Load** options to have indexes added after the load.

There might be times when you want to select **Leave Off** in the **After Load** field. Here is an example of a situation when you might want to leave the indexes off during and after the table loading for performance reasons. The table is updated multiple times in a series of load steps that appear in separate jobs. Indexes are defined on the table only to improve performance of a query and reporting application that runs after the nightly load. None of the load steps need the indexes, and leaving the indexes on impedes the performance of the load. In this scenario, the indexes can be taken off before the first update and left off until after the final update.

Considering a Bulk Load

Problem

You want to load a table on a row-by-row basis, but you find that it takes too long to load.

Solution

You should consider using the optimized SAS/ACCESS engine bulk loaders to bulk load the data into database tables. Bulk-load options are set in the metadata for an RDBMS library. To set these options from the SAS Data Integration Studio Inventory tree, right-click an RDBMS library, then select **Properties ▶ Options ▶ Advanced Options ▶ Output**. Select the check box that will enable the RDBMS bulk-load facility for the current library.

You can set additional bulk-load options for the tables in an RDBMS library. To set these options from the SAS Data Integration Studio Inventory tree, right-click an RDBMS table, then select **Properties ▶ Physical Storage ▶ Table Options**. Specify the appropriate bulk-load option for the table.

Also, you should consider using native SAS/ACCESS engine libraries instead of ODBC libraries or OLEDB libraries for RDBMS data.

CHAPTER

16 Working with SAS Sorts

About SAS Sort Transformations

The SAS Sort transformation provides a graphic interface for the functions available in PROC SORT. You can use the transformation to read data from a source, sort it, and write the sorted data to a target in a SAS Data Integration Studio job. Sorting occurs implicitly with index creation, ORDER BY clauses, SAS SQL joins, and procedure execution that requires ordering. For any caller, the underlying sort engine is the same. Sort callers include, but are not limited to, the DATA step, PROC SORT, PROC SQL, and PROC SUMMARY.

The properties window for the SAS Sort transformation contains tabs that enable you to select the columns that you sort by and to set options for the sort, as described in "Setting Sort Options" on page 276. You can also optimize sort performance, as described in "Optimizing Sort Performance" on page 277. For an example of how you can use a SAS Sort transformation, see "Creating a Table That Sorts the Contents of a Source" on page 279.

Setting Sort Options

Problem

You want to set options for a table sort. The sort is performed with the SAS Sort transformation in a SAS Data Integration Studio job.

Solution

You can access the **`Options`** tab in the properties window of the SAS Sort transformation. The available options are listed in the following table.

Table 16.1 SAS Sort Options

Option	Description
Create SYSLAST Macro Variable	Specifies whether SAS Data Integration Studio generates a SYLAST macro statement at the end of the current transformation. Accept the default value of **`YES`** when the current transformation creates an output table that should be the input of the next transformation in the process flow. Otherwise, select **`NO`**.
Equals	Maintains relative order within BY groups. The following values are available: □ NOEQUALS □ EQUALS (default)
Force	Overrides the default sort behavior by forcing data to be sorted, even if the information that is stored with the table indicates that the data is already in sorted order. The following values are available: □ FORCE □ no FORCE (default)
Tagsort	Reduces temporary disk usage. The following values are available: □ TAGSORT □ no TAGSORT (default)
Duplicates	Deletes duplicate observations. The following values are available: □ Blank/no value (default) □ NODUPKEY □ NODUPREC

Option	Description
Sortseq	Specifies the collating sequence. The following values are available: ☐ ASCII ☐ DANISH (alias NORWEGIAN) ☐ EBCDIC ☐ FINNISH ☐ ITALIAN ☐ NATIONAL ☐ REVERSE ☐ SPANISH ☐ SWEDISH ☐ User-supplied
Sortsize	Specifies the maximum amount of memory that is available to PROC SORT. The following values are available: ☐ MAX ☐ MIN ☐ n ☐ nK ☐ nM ☐ nG
System Options	Specifies one or more SAS System options on the OPTIONS statement. For details, see *SAS Language Reference: Dictionary*.
PROC SORT Options	Specifies one or more options in the SORT procedure.

Optimizing Sort Performance

Problem

You want to sort the data in your source tables before running a job. Sorting is a common and resource-intensive component of SAS Data Integration Studio. Sorts occur explicitly as PROC SORT steps and implicitly in other operations such as joins. Effective sorting requires a detailed analysis of performance and resource usage.

Sorting large SAS tables requires large SORT procedure utility files. When SAS Data Integration Studio is running on multiple SAS jobs simultaneously, multiple SORT procedure utility files can be active. For these reasons, tuning sort performance and understanding sort disk space consumption are critical.

Solution

You can enhance sort performance with the techniques listed in the following table. For more information, see the ETL Performance Tuning Tips whitepaper that is available from **`http://support.sas.com/documentation/whitepaper/technical/`**.

Table 16.2 Sort Performance Enhancement Techniques

Technique	Notes
Use the improved SAS®9 sort algorithm	SAS®9 includes a rewritten SORT algorithm that incorporates threading and data latency reduction algorithms. The SAS®9 sort uses multiple threads and outperforms a SAS 8 sort in almost all circumstances.
Minimize data	Perform the following steps: □ Minimize row width. □ Drop unnecessary columns. □ Minimize pad bytes.
Direct sort utility files to fast storage devices	Use the WORK invocation option, the UTILLOC invocation option, or both options to direct SORT procedure utility files to fast, less-utilized storage devices. Some procedure utility files are accessed heavily, and separating them from other active files might improve performance.
Distribute sort utility files across multiple devices	Distribute SORT procedure utility files across multiple fast, less-utilized devices. Direct the SORT procedure utility file of each job to a different device. Use the WORK invocation option, the UTILLOC invocation option, or both options.
Pre-sort explicitly on the most common sort key	SAS Data Integration Studio might arrange a table in sort order, one or multiple times. For large tables in which sort order is required multiple times, look for a common sort order. Use the MSGLEVEL=I option to expose information that is in the SAS log to determine where sorts occur.
Change the default SORTSIZE value	For large tables, set SORTSIZE to 256 MB or 512 MB. For extremely large tables (a billion or more wide rows), set SORTSIZE to 1 GB or higher. Tune these recommended values further based on empirical testing or based on in-depth knowledge of your hardware and operating system.
Change the default MEMSIZE value	Set MEMSIZE at least 50% larger than SORTSIZE.
Set the NOSORTEQUALS system option	In an ETL process flow, maintaining relative row order is rarely a requirement. If maintaining the relative order of rows with identical key values is not important, set the system option NOSORTEQUALS to save resources.
Set the UBUFNO option to the maximum of 20	The UBUFNO option specifies the number of utility I/O buffers. In some cases, maximizing UBUFNO increases sort performance up to 10%. Increasing UBUFNO has no negative ramifications.

Technique	Notes
Use the TAGSORT option for nearly sorted data	TAGSORT is an alternative SAS 8 sort algorithm that is useful for data that is almost in sort order. The option is most effective when the sort-key width is no more than 5 percent of the total uncompressed column width. Using the TAGSORT option on a large unsorted data set results in extremely long sort times compared to a SAS®9 sort that uses multiple threads.
Use relational database sort engines to pre-sort tables without data order issues	Pre-sorting in relational databases might outperform sorting based on SAS. Use options of the SAS Data Integration Studio Extract transformation to generate an ORDER BY clause in the SAS SQL. The ORDER BY clause asks the relational database to return the rows in that particular sorted order.
Determine disk space requirements to complete a sort	Size the following sort data components: ☐ Input data ☐ SORT procedure utility file ☐ Output data
Size input data	Because sorting is so I/O intensive, it is important to start with only the rows and columns that are needed for the sort. The SORT procedure WORK files and the output file will be dependent on the input file size.
Size SORT procedure utility files	Consider a number of factors to size the SORT procedure utility files: ☐ sizing information of the input data ☐ any pad bytes added to character columns ☐ any pad bytes added to short numeric columns ☐ pad bytes that align each row by 8-bytes (for SAS data sets) ☐ 8 bytes per row overhead for EQUALS processing ☐ per-page unused space in the SORT procedure utility files ☐ multi-pass merge: doubling of SORT procedure utility files (or sort failure)
Size of output data	To size the output data, apply the sizing rules of the destination data store to the columns that are produced by the sort.

Creating a Table That Sorts the Contents of a Source

Problem

You want to create a job that reads data from a source, sorts it, and writes the sorted data to a target.

Solution

You can create a job that uses a SAS Sort transformation to sort the data in a source table and write it to a target table.

Tasks

Create and Populate the Job

Perform the following steps to create and populate a new job:

1 Create a new job.

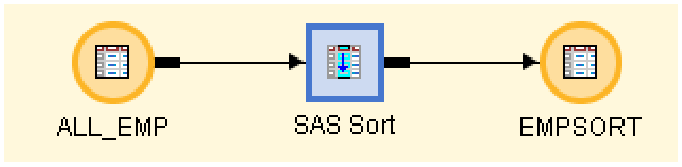
2 Select and drag the SAS Sort transformation from the Data Transforms folder in the Process Library tree into the empty job in the Process Designer window.

3 Drop the source table on the source drop zone for the SAS Sort transformation.

4 Drop the target table on the target drop zone for the SAS Sort transformation.

5 Delete the Table Loader and the temporary worktable from the job. The job resembles the sample shown in the following display.

Display 16.1 Sample SAS Sort Process Flow Diagram



Specify How Information in the Target Is to Be Sorted and Run the Job

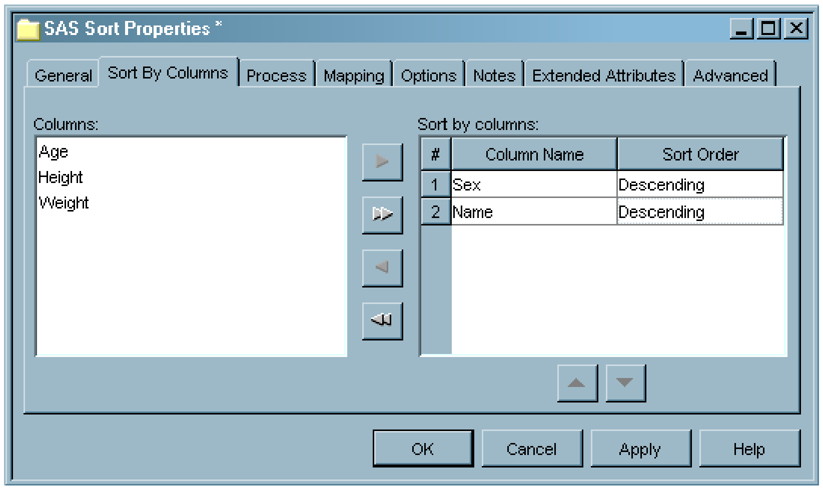
Perform the following steps to specify how information in the target table is to be sorted:

1 Open the **Sort By Columns** tab of the properties window for the SAS Sort transformation.

2 Select the first variable for the new sort from the list in the **Columns** field. Then, move the variable to the **Sort by columns** field. Then, specify the sort direction for the variable with the drop-down menu in the Sort Order column.

3 Move the other variables that you want to sort by to the **Sort by columns** field. Then, set the sort direction for each. The following display depicts the completed **Sort By Columns** tab for the sample sort job.

Display 16.2 Completed SAS Sort Tab for Sample Job



Columns:
Age
Height
Weight

Sort by columns:

#	Column Name	Sort Order
1	Sex	Descending
2	Name	Descending

4 Save the selection criteria for the target and close the properties window.

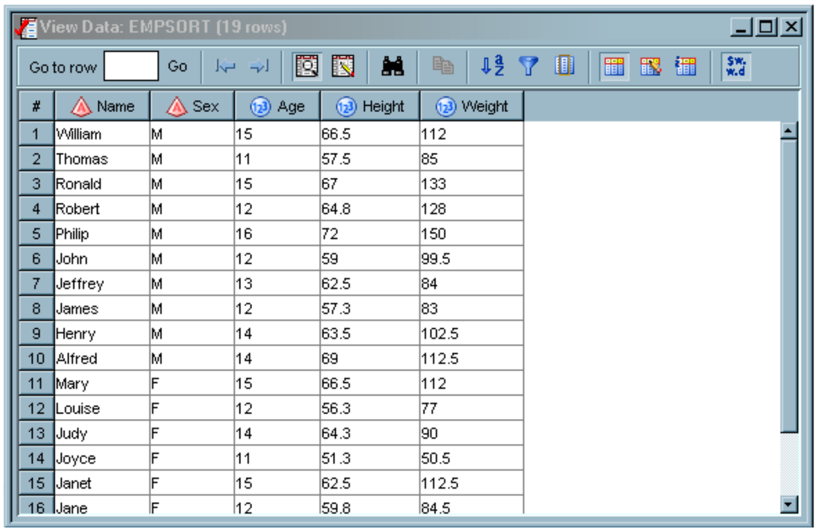
5 Run the job. If you are prompted to do so, enter a user ID and password for the default SAS Application Server that generates and run SAS code for the job. The server executes the SAS code for the job.

6 If the job completes without error, go to the next section. If error messages appear, read and respond to the messages.

View the Output

You can verify that the job created the desired output by reviewing the View Data window. The View Data window for the sample job is shown in the following display.

Display 16.3 Data in Sample Sorted Table



#	Name	Sex	Age	Height	Weight
1	William	M	15	66.5	112
2	Thomas	M	11	57.5	85
3	Ronald	M	15	67	133
4	Robert	M	12	64.8	128
5	Philip	M	16	72	150
6	John	M	12	59	99.5
7	Jeffrey	M	13	62.5	84
8	James	M	12	57.3	83
9	Henry	M	14	63.5	102.5
10	Alfred	M	14	69	112.5
11	Mary	F	15	66.5	112
12	Louise	F	12	56.3	77
13	Judy	F	14	64.3	90
14	Joyce	F	11	51.3	50.5
15	Janet	F	15	62.5	112.5
16	Jane	F	12	59.8	84.5

You can review the View Data window to ensure that the data from the source table was properly sorted. Note that the Name and Sex columns in the sample target table are sorted, but the other columns remained unsorted.

CHAPTER
17

Working with the SQL Join Transformation

About SQL Join Transformations

The SQL Join transformation enables you to create SQL queries that run in the context of SAS Data Integration Studio jobs. The transformation features a graphical interface that provides a consistent and intuitive setting for building the statements and clauses that constitute queries. The transformation supports the PROC SQL syntax of **`Create table/view <table> as <query expression>`** and accommodates up to 256 tables in a single query.

The Select statement now supports joining the table to itself. It also supports subqueries; the CASE expression; and WHERE, GROUP BY, HAVING, and ORDER BY clauses. Finally, the current version of the SQL Join transformation inserts a comment into the generated code for queries that states that the code was generated by version 2 of the transformation. This comment distinguishes the code from the code that is generated by the earlier version of the SQL Join transformation.

The process of building the SQL query is performed on the **Designer** tab. Use the **Designer** tab to create, edit, and review an SQL query. The tab contains sections that are designed to simplify creating the SQL query and configuring its parts.

Using the SQL Designer Tab

Problem

You want to create SQL queries that you can use in SAS Data Integration Studio jobs. You want to build these queries in a graphical interface that enables you to drag and drop components onto a visual representation of a query. After a component is added to the query, you need the ability to open and configure it.

Solution

Use the **Designer** tab of the properties window for the SQL transformation to create, edit, and review an SQL query. The tab contains sections that are designed to simplify creating the SQL query and configuring its parts.

Tasks

Using Components on the Designer Tab

The **Designer** tab enables you to perform the tasks listed in the following table:

Table 17.1 Designer Tab Tasks

If you want to	Use the	To access
Select and manipulate an object that displays in the **Create** tab.	Navigate pane	Click the object that you need to access.
Add SQL clauses to the flow shown on the **Create/Subquery** tab.	SQL Clauses pane	Double-click the clause or drop it on the **Create** tab.
Review the list of columns in the source table and the target table. Note that you can specify alphabetic display of the columns by selecting **Display columns in alphabetical order**.	Tables pane	Click **Select**, **Where**, **Having**, **Group by**, or **Order by** in the SQL Clauses pane.
Display and update the main properties of an object that is selected on the **Create** tab. The title of this pane changes to match the object selected in the Navigate pane.	Properties pane	Click an object on the **Create** tab.

If you want to	Use the	To access
Create SQL statements, configure the clauses contained in the statement, and edit the source table to target table mappings. The name of this component changes as you click different statements and clauses in the Navigate pane.	**`Create`** tab	Click **`Create`** in the Navigate pane.
View the SAS code generated for the query.	**`Source`** tab	Click **`Source`** at the bottom of the **`Create`** tab.
View the log of a SAS program, such as the code that is executed or validated for the SQL query.	**`Log`** tab	Click **`Log`** at the bottom of the **`Create`** tab.

Additional Information

For more information about using the Navigate pane, see the "Navigate Pane" topic in SAS Data Integration Studio help. For information about using the **`Create Query/Subquery`** tab, see the "Create/Subquery Tab" topic. For information about other **`Designer`** tab components, see the "Where/Having/Join Tab" topic, the "Group by Tab" topic, the "Order By Tab" topic, and the "Select Tab" topic. For information about the **`Source`** and **`Log`** tabs, click **`Help`** when the component is displayed.

Reviewing and Modifying Clauses, Joins, and Tables in an SQL Query

Problem

You want to view a clause, join, or table in an SQL query or modify its properties.

Solution

Use the Navigate and properties panes on the **`Designer`** tab of the properties window for the SQL transformation. If you click an SQL clause or a join, its SQL code is highlighted in the Source pane. If you change a property for a clause, join, or table in its properties pane, the change is also displayed in the Source pane.

Tasks

Review Clauses, Join, and Tables

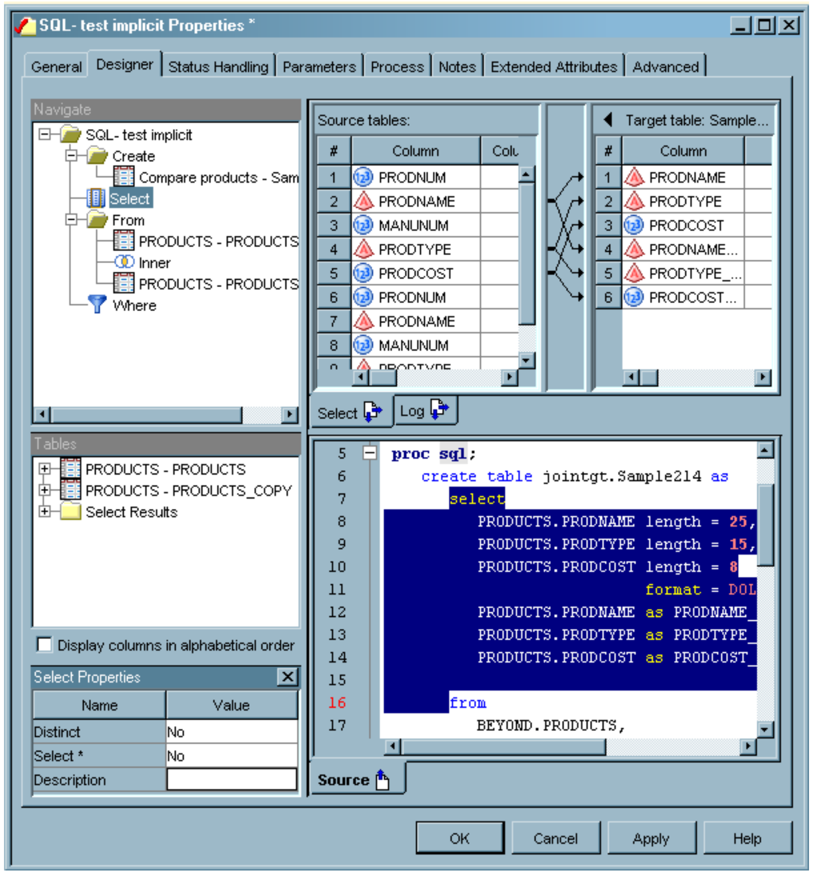
When you click an item in the Navigate pane, the **`Designer`** tab responds in the following ways:

- The properties pane for the clause, join, or table is displayed.

- □ The appropriate tab for the clause or join is displayed on the right side of the **Designer** tab. When you click a table, the currently displayed tab continues to be shown.
- □ The appropriate SQL code is highlighted on the **Source** tab when you click a clause or a join. The **Source** tab is unaffected when you click a table.
- □ If you click SQL Join, Create, or From in the Navigate pane, the SQL Clauses pane is displayed.
- □ If you click Select, Where, or one of the Joins in the Navigate pane, the Tables pane is displayed.

The following display shows the **Designer** tab for a sample job.

Display 17.1 Information About a Select Clause on a Designer Tab



#	Column	Colu
1	PRODNUM	
2	PRODNAME	
3	MANUNUM	
4	PRODTYPE	
5	PRODCOST	
6	PRODNUM	
7	PRODNAME	
8	MANUNUM	

#	Column
1	PRODNAME
2	PRODTYPE
3	PRODCOST
4	PRODNAME...
5	PRODTYPE_...
6	PRODCOST...

```
proc sql;
   create table jointgt.Sample214 as
      select
         PRODUCTS.PRODNAME length = 25,
         PRODUCTS.PRODTYPE length = 15,
         PRODUCTS.PRODCOST length = 8
                         format = DOL
         PRODUCTS.PRODNAME as PRODNAME_
         PRODUCTS.PRODTYPE as PRODTYPE_
         PRODUCTS.PRODCOST as PRODCOST_

      from
         BEYOND.PRODUCTS,
```

Select Properties

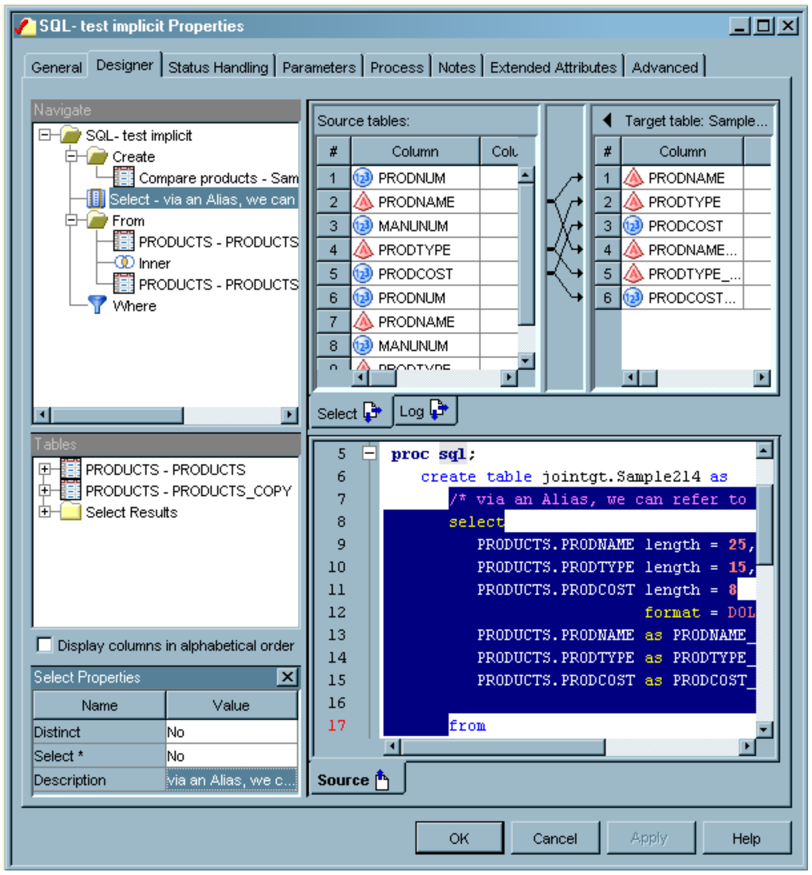
Name	Value
Distinct	No
Select *	No
Description	

Note that **Select** is highlighted on the Navigate pane, and the SQL code for the SELECT clause is highlighted on the **Source** tab. Also note that the **Select** tab, the Tables pane, and the Select Properties pane are displayed.

Modify Properties of Clauses and Tables

You can use the properties pane that is displayed when you click an object on the Navigate pane as if it were the **Options** tab in the stand-alone properties window for the object. For example, if you enter text in the **Description** field in the Select Properties pane, a comment is added to the SELECT clause on the **Source** tab. See the following display for a sample view of this behavior.

Display 17.2 Using the Description Field to Comment a Select Clause



Select Properties

Name	Value
Distinct	No
Select *	No
Description	via an Alias, we c...

Source tables:

#	Column	Colu
1	PRODNUM	
2	PRODNAME	
3	MANUNUM	
4	PRODTYPE	
5	PRODCOST	
6	PRODNUM	
7	PRODNAME	
8	MANUNUM	
9	PRODTYPE	

Target table: Sample...

#	Column
1	PRODNAME
2	PRODTYPE
3	PRODCOST
4	PRODNAME...
5	PRODTYPE_...
6	PRODCOST...

```
proc sql;
   create table jointgt.Sample214 as
      /* via an Alias, we can refer to
      select
         PRODUCTS.PRODNAME length = 25,
         PRODUCTS.PRODTYPE length = 15,
         PRODUCTS.PRODCOST length = 8
                           format = DOL
         PRODUCTS.PRODNAME as PRODNAME_
         PRODUCTS.PRODTYPE as PRODTYPE_
         PRODUCTS.PRODCOST as PRODCOST_

      from
```

Note that text entered in the **Description** field in the Select Properties pane is also displayed next to the Select object in the Navigate pane and immediately before the SQL code on the **Source** tab. If you were to delete the text from the **Description** field, it would also be removed from the Navigate pane and the **Source** tab. You can make similar modifications to any field in a properties pane for any object, unless the field is dimmed. Dimmed fields are read-only.

Understanding Automatic Joins

The automatic join (auto-join) process determines the initial relationships and conditions for a query formulated in the SQL Join transformation. You can understand how these relationships and conditions are established. You can also examine how drop order, key relationships, and indexes are used in the auto-join process.

The process for determining the join relationships is based on the order of the tables added to SQL transformation as input. When more than one table is dropped on the SQL transformation, a best guess is made about the join relationships between the tables. The join order is determined by taking the first table dropped and making it the left side of the join. Then, the next table dropped becomes the right side. If more than two tables are dropped, the next join is added so that the existing join is placed on the left side and the next table is placed on the right. This process continues until no more source tables are found. The default join type is an inner join.

As each join is created and has its left and right sides added, a matching process is run to determine the best relationships for the join. The process evaluates the join tables from the left side to the right side. For example, if a join is connected on the left, it follows that left side join until it hits all of the tables that are connected to the join. This includes any joins that are connected to it.

The auto-join process is geared toward finding the best relationships based on the known relationships documented in the physical tables when each of the tables being joined contains key constraints, indexes, or both. Therefore, the process is most likely to find the correct relationships when the primary and foreign key relationships are defined between the tables that are being joined. The auto-join process can still find the correct relationships using indexes alone, but an index-only match can occur only when there are columns that are matched between the two tables in the join.

The key-matching process proceeds as follows:

1 Each of the left side table's unique keys are evaluated to find any existing associated foreign keys in any table on the right side of the join. If no associations are found, the left side table's foreign keys are checked to see whether a relationship is found to a unique key in a table on the right side of the join. If a match is found, both tables are removed from the search.

2 If tables are still available on both the left and right sides, the table indexes are searched. The left side is searched first. If an index is found, then the index columns are matched to any column in the tables on the right. As matches are found, both tables are removed from the search. The right side is searched if tables are still available on both the right and left sides.

3 If tables are still available on both the left and right sides, the left side table's columns are matched to the right side by name and type. If the type is numeric, the lengths must match. As a match is found, both tables are removed from the search.

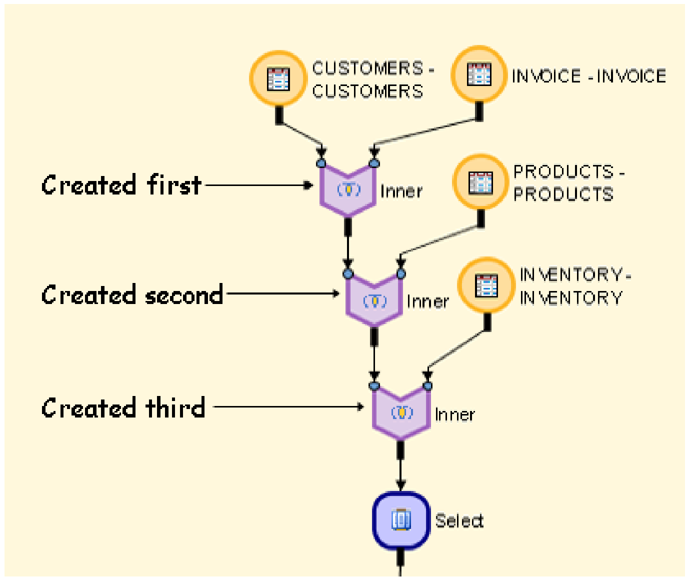
A Sample Auto-Join Process

This is an abstract explanation. Therefore, it is useful to illustrate it with a specific example. Suppose that you add the following tables as input to the SQL Join transformation in the following order:

- CUSTOMERS, with the following constraint defined:
  - Primary key: CUSTOMER_ID
- INVOICE, with the following constraints defined:
  - Primary key: INVOICE_NUMBER
  - Foreign key: CUSTOMER_ID
  - Foreign key: PRODUCT_NUMBER
- PRODUCTS, with the following constraint defined:
  - Primary key: PRODUCT_NUMBER
- INVENTORY, with the following constraint defined:
  - Index: PRODUCT_NUMBER

After the auto-join process is run for this source data, the process flow depicted in the following display is shown in the **`Create`** tab in the **`Designer`** tab in the properties window for the SQL Join transformation.

Display 17.3 Sample Process Flow for an Auto-Join Process



This process flow is resolved to the following order: CUSTOMERS, INVOICE, PRODUCTS, and INVENTORY. As each join is created and has its left and right sides, a matching process is used to determine the best relationships for the join. The process evaluates the join tables from the left side to the right side. For example, if a join is connected on the left, it follows that left side join until it gets all of the tables that are connected to the join. The matching process uses the following criteria to determine a good match. Note that the tables are removed from the search process as the relationships are found.

The first join is created with the left table of CUSTOMERS and the right table of INVOICE. Going through the join relationship process, the key relationship on CUSTOMER_ID is found between the two tables. Both tables are removed from the search and the matching process is finished.

The next join is created with the search results of the CUSTOMERS and INVOICE tables as the new left table and PRODUCTS as the right table. A key relationship between INVOICE and PRODUCTS on the column PRODUCT_NUMBER is found, and an expression is created. Both tables are removed from the search and the matching process is finished.

The last join is created with the search results of the CUSTOMER, INVOICE, and PRODUCTS table as the new left table and INVENTORY as the right table. No key relationships are found, so the indexes are searched. A match is found between PRODUCTS and INVENTORY on the column PRODUCT_NUMBER. Both tables are then removed from the search and the matching process is finished.

The relationship is initialized as follows:

```
CUSTOMERS.CUSTOMER_ID = INVOICE.CUSTOMER_ID and
INVOICE.PRODUCT_NUMBER = PRODUCTS.PRODUCT_NUMBER and
PRODUCTS.PRODUCT_NUMBER = INVENTORY.PRODUCT_NUMBER
```

Selecting the Join Type

Problem

You want to select a specific type for a join in an SQL query. You can use the join type selection to gain precise control over the data included in the results of the query.

Solution

Right-click an existing join in an SQL query, and click the appropriate join type in the pop-up menu.

Tasks

Change Join Types in a Sample SQL Query

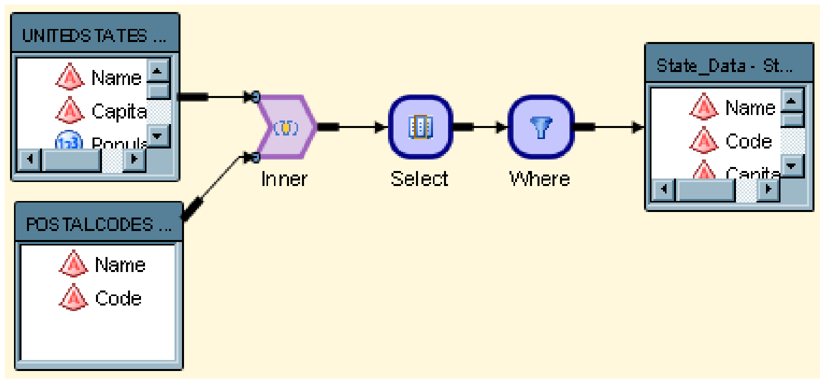
Examine a sample SQL query in a SAS Data Integration Studio job to see the effects of changing the join types used in the query. The sample query contains the tables and columns listed in the following table:

Table 17.2 Sample Query Data

Source Table 1: POSTALCODES	**Source Table 2: UNITEDSTATES**	**Target Table: State_Data**
□ Name	□ Capital	□ Name
□ Code	□ Population	□ Code
	□ Area	□ Capital
	□ Continent	□ Population
	□ Statehood	□ Area
		□ Continent
		□ Statehood

The join condition for the query is POSTALCODES.Name = UNITEDSTATES.Name. The query is depicted in the following display.

Display 17.4 Sample SQL Query in a SAS Data Integration Studio Job



Notice that the query contains an inner join and a WHERE statement. These components are included by default when a query is first created. The following table illustrates how the query is affected when you run through all of the available join types in succession:

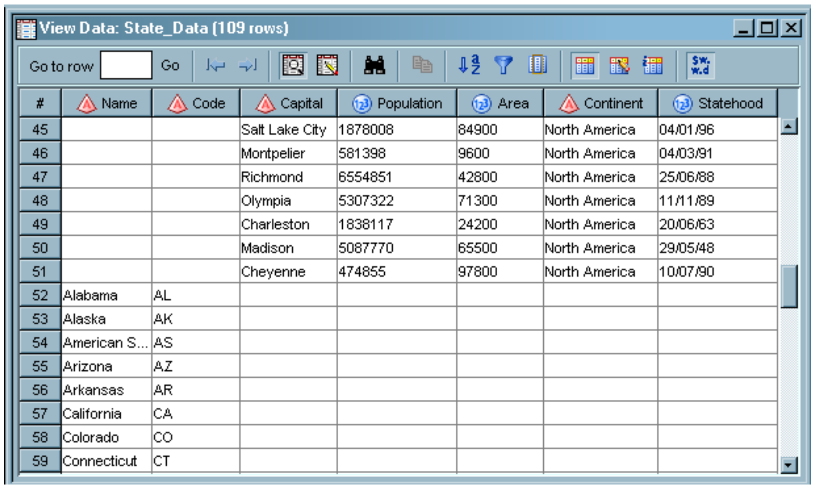
Table 17.3 Results By Join Type

Join Type	Description	Data Included in Results	Implicit/ Explicit Status
Inner	Combines and displays only the rows from the first table that match rows from the second table, based on the matching criteria that are specified in the WHERE clause.	50 rows: 50 matches on name column; 0 non-matches	Implicit
Full	Retrieves both the matching rows and the non-matching rows from both tables.	59 rows: 50 matches on name column; 8 non-matches from POSTALCODES (left table); 1 non-match from UNITEDSTATES (right table)	Explicit
Left	Retrieves both the matching rows and the non-matching rows from the left table.	58 rows: 50 matches on name column; 8 non-matches from POSTALCODES (left table)	Explicit
Right	Retrieves both the matching rows and the non-matching rows from the right table.	51 rows: 50 matches on name column; 1 non-match from UNITEDSTATES (right table)	Explicit

Join Type	Description	Data Included in Results	Implicit/ Explicit Status
Cross	Combines each row in the first table with every row in the second table (creating a Cartesian product of the tables).	2958 rows	Explicit
Union	Selects unique rows from both tables together and overlays the columns. PROC SQL first concatenates and sorts the rows from the two tables, and then eliminates any duplicate rows. See the following display for the results of a sample union join.	109 rows: 58 rows from POSTALCODES (left table); 51 rows from UNITEDSTATES (right table)	Explicit

A section of the View Data window for a sample query that includes a union join is depicted in the following display.

Display 17.5 Sample Section from a View of a Union Join



#	Name	Code	Capital	Population	Area	Continent	Statehood
45			Salt Lake City	1878008	84900	North America	04/01/96
46			Montpelier	581398	9600	North America	04/03/91
47			Richmond	6554851	42800	North America	25/06/88
48			Olympia	5307322	71300	North America	11/11/89
49			Charleston	1838117	24200	North America	20/06/63
50			Madison	5087770	65500	North America	29/05/48
51			Cheyenne	474855	97800	North America	10/07/90
52	Alabama	AL					
53	Alaska	AK					
54	American S...	AS					
55	Arizona	AZ					
56	Arkansas	AR					
57	California	CA					
58	Colorado	CO					
59	Connecticut	CT					

Rows 45 to 51 come from the POSTALCODES table. Rows 52 to 59 come from the UNITEDSTATES table.

These joins are contained in the FROM clause in the SELECT statement, which comes earlier in an SQL query than a WHERE statement. You can often create more efficient query performance by using the proper join type in a SELECT statement than you can by setting conditions in a WHERE statement that comes later in the query.

Adding User-Written SQL Code

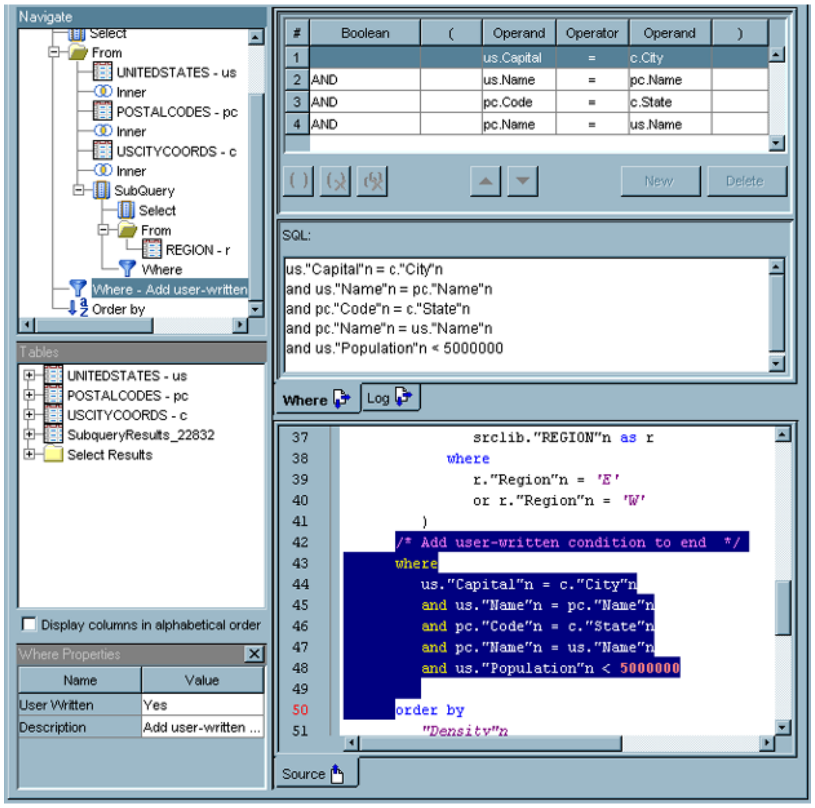
Problem

You want to add user-written code to an SQL query that is used in a SAS Data Integration Studio job. This user-written code can consist of SQL code that is added to a WHERE, HAVING, or JOIN clause. It can also overwrite the entire DATA step for the SQL Join transformation.

Solution

You can add SQL code to an SQL WHERE, HAVING, or JOIN clause in the properties window for the clause. To set the user-written property for a clause, click the clause in the SQL Clauses pane on the **`Designer`** tab for the SQL Join transformation. Then, select **`Yes`** in the **`User Written`** field and enter the code in the **`SQL`** field on the clause's tab. The following display shows sample user-written code added to a WHERE clause.

Display 17.6 Sample User-Written SQL Code



#	Boolean	(	Operand	Operator	Operand	)
1			us.Capital	=	c.City	
2	AND		us.Name	=	pc.Name	
3	AND		pc.Code	=	c.State	
4	AND		pc.Name	=	us.Name	

```
us."Capital"n = c."City"n
and us."Name"n = pc."Name"n
and pc."Code"n = c."State"n
and pc."Name"n = us."Name"n
and us."Population"n < 5000000
```

```
                    srclib."REGION"n as r
                 where
                    r."Region"n = 'E'
                    or r."Region"n = 'W'
              )
           /* Add user-written condition to end  */
           where
              us."Capital"n = c."City"n
              and us."Name"n = pc."Name"n
              and pc."Code"n = c."State"n
              and pc."Name"n = us."Name"n
              and us."Population"n < 5000000

           order by
              "Density"n
```

Name	Value
User Written	Yes
Description	Add user-written ...

Note that the following line of SQL code was added to the **`SQL`** field on the **`Where`** tab:

```
and us."Population"n < 5000000
```

This code is also highlighted on the **`Source`** tab.

Additional Information

For information about how to overwrite the entire DATA step for the SQL Join transformation, see Chapter 12, “Working with User-Written Code,” on page 215.

Debugging an SQL Query

Problem

You want to determine which join algorithm is selected for an SQL query by the SAS SQL Optimizer. You also need to know how long it takes to run the job that contains the SQL Join transformation.

Solution

You can enable debugging for the query by setting the **Debug** property in the SQL Properties pane.

Tasks

Set the Debug Property

The **Debug** property in the SQL Properties pane enables the following debugging option:

```
options sastrace = ',,,sd' sastraceloc = saslog no$stsuffix fullstimer;
```

You can use this option to determine which join algorithms are used in the query and to get timing data for the SAS job.

You can use the keywords from the trace output that are listed in the following table to determine which join algorithm was used:

Table 17.4 Debugging Keywords and Join Algorithms

Keyword	Join Algorithm
sqxsort	sort step
sqxjm	sort-merge join
sqxjndx	index join
sqxjhsh	hash join
sqxrc	table name

Examine Some Sample Method Traces

The following sample fragments illustrate how these keywords appear in a _method trace.

In the first example, each data set is sorted and sort-merge is used for the join:

```
sqxjm
    sqxsort
        sqxsrc( WORK.JOIN_DATA2 )
    sqxsort
        sqxsrc( LOCAL.MYDATA )
```

In the next example, an index nested loop is used for the join:

```
sqxjndx
    sqxsrc( WORK.JOIN_DATA2 )
    sqxsrc( LOCAL.MYDATA )
```

In the final example, a hash is used for the join:

```
sqxjhsh
    sqxsrc( LOCAL.MYDATA )
    sqxsrc( WORK.JOIN_DATA1 )
```

Adding a Column to the Target Table

Problem

You want to add a column to the target table for an SQL query that is used in a SAS Data Integration Studio job.

Solution

You can use the **`Columns`** tab on the properties window for the target table to add a column to the target table. (You can also add a column in the **`Select`** tab. To do this, right-click in the **`Target table`** field and click **`New Column`** in the pop-up menu.)

Tasks

Add a Column with the Columns Tab for the Target Table

Perform the following steps to add a column to the target table:

1. Right-click the target table in the Navigation pane. Then, open the **`Columns`** tab in its properties window.
2. Click **`New`** to add a row to the list of columns.
3. Enter the column name in the **`Column`** field of the new row.
4. Click the drop-down menu in the **`Type`** field. Then, click either **`Character`** or **`Numeric`**.
5. Review the other columns in the new row to ensure that they contain appropriate values. Make any needed changes.
6. Click **`OK`** to save the new column and close the properties window.

Adding a Join to an SQL Query in the Designer Tab

Problem

You want to add a join to an SQL query that is used in a SAS Data Integration Studio job. Then you can connect an additional source table, join, or subquery for the query to the join.

Solution

You can drop the join on the **Create** tab in the **Designer** tab. The **Designer** tab is located in the properties window for the SQL Join transformation. This join enables you to add a new drop zone to the query flow in the Create pane.

Tasks

Add a Join to the Create Tab

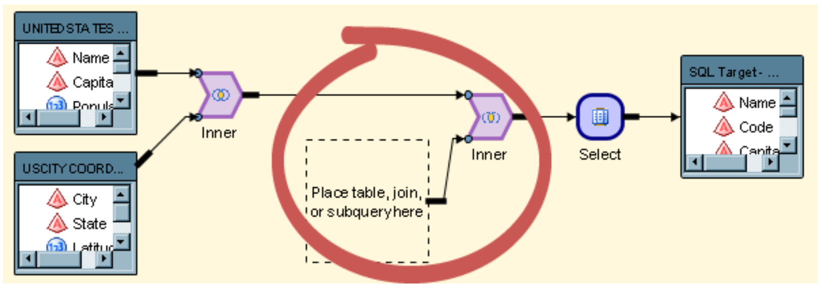
Perform the following steps to add a join to the **Create** tab:

1 Select one of the join objects in the Joins folder in the SQL Clauses pane, and drop it on the **Select** object in a query flow displayed on the **Create** tab. The join object and its drop zone are added to the query flow. (You can also double-click the join object to insert it between the existing join and the SELECT clause.)

2 Drop an appropriate table, join, or SubQuery object on the drop zone. The SQL code for the join is shown on the **Source** tab.

The join and its drop zone are displayed in the query flow, as shown in the following display.

Display 17.7 Add a Join and Drop Zone



Note: You can add the source and target tables directly to the process flow diagram for the job. You can also add a table, join, or subquery to a job by dragging and dropping it on an empty source or target drop zone on the **Create** tab on the **Designer** tab of the properties window for the SQL Join transformation. If you drop a table on an existing table on the **Designer** tab, the new table will replace the existing table. △

Creating a Simple SQL Query

Problem

You want to add a simple SQL query to a SAS Data Integration Studio job.

Solution

Use the SQL Join transformation. The SQL Join transformation enables you to create SQL queries that run in the context of SAS jobs. The transformation features a graphical interface that enables you to build the statements and clauses that constitute queries. This example describes how to use the transformation to create a job that uses an SQL query to select data from two SAS tables. The data is merged into a target table.

Tasks

Create and Populate the Job

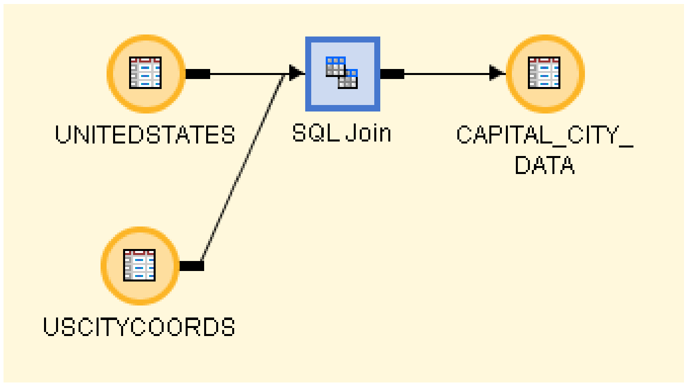
Perform the following steps to create and populate the job:

1. Create an empty job.
2. Drop the SQL Join transformation from the Data Transformations folder in the Process Library tree onto the empty job.
3. Drop the first source table on the input drop zone for the SQL Join transformation.
4. Drop any additional source tables on the SQL Join transformation.
5. Drop the target table on the target drop zone.
6. Delete the Table Loader transformation and the temporary worktable SQL Target from the job. If you keep the Table Loader and the worktable, you must configure two sets of mappings, one from the source tables to the worktable and another from the worktable to the target table. The extra processing required could degrade performance when the job is run. In addition, the Table Loader step should be deleted if you use pass-through processing when your target table is a DBMS table and your DBMS engine supports the **`Create as Select`** syntax.

Note: If you use work tables as the source for your query, you can use the **`Run Prior Steps on Submit`** option to control whether the steps that are used to create the source tables for the SQL query are run for a given query submission. When you select the option, the steps that are placed before the SQL query code are run each time that the query is submitted. These steps are not needed when you want to check only the results of the query. After that first run, you can deselect the option to improve your performance while you develop and test the query within the **`Designer`** tab. However, you must run the query at least once to generate the source tables. △

The following display shows a sample SQL job.

Display 17.8 Sample SQL Job



Now you can create the SQL query that populates the target table.

Create the SQL Query

Perform the following steps to create the SQL query that populates the target table:

1 Open the properties window for the SQL Join transformation.

2 Click **`Designer`** to access the **`Designer`** tab.

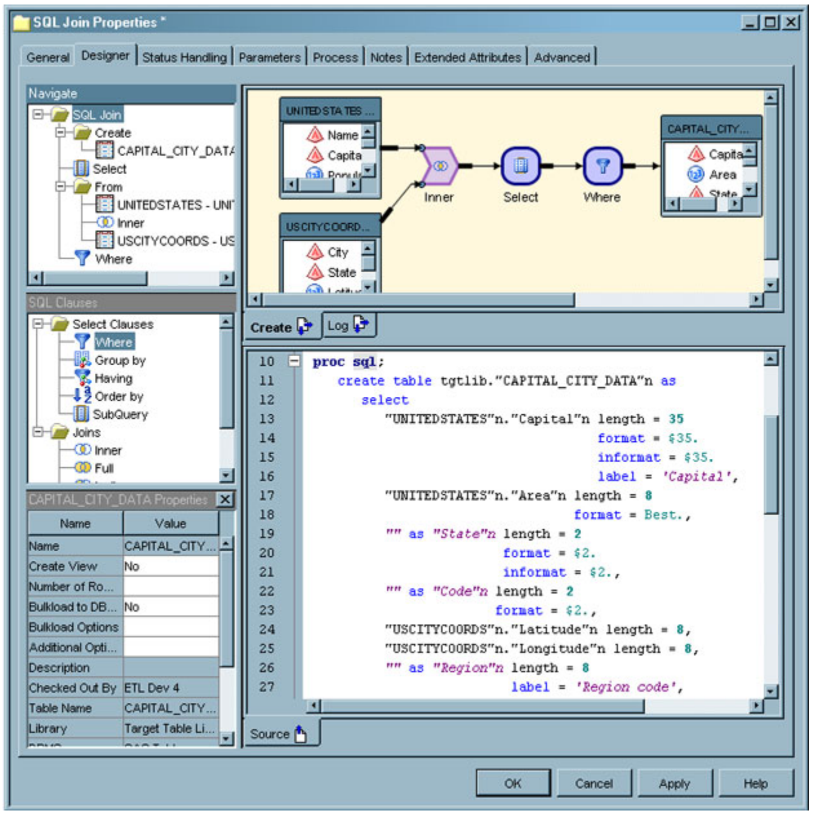
3 Click **`SQL Join`** in the Navigate pane. The right-hand side of the **`Designer`** tab contains a Navigate pane, an SQL Clauses/Tables pane, and a properties pane. You might need to resize the horizontal borders of the panes to see all three of them. For more information, see "Using the SQL Designer Tab" on page 286.

   You can enter options that affect the entire query. Note that the SQL Join Properties pane displays at the bottom of the tab. For example, you can limit the number of observations output from the job in the **`Max Output Rows`** field.

4 Click **`Create`** in the Navigate pane to display an initial view of the query on the **`Create`** tab. Note that the sample query already contains an INNER join, a SELECT clause, and a WHERE clause. These elements are created when you drop source tables on the transformation template. The joins shown in the query process flow are not necessarily joined in the order in which the SQL optimizer will actually join the tables. However, they do reflect the SQL syntax.

   The **`Show Columns`** option for the query flow diagram on the **`Create`** tab is also turned on. This option provides scrollable lists of the columns included in each of the source and target tables for the job, but the time that it takes to generate the column lists can degrade performance. If you need to improve performance, you can right-click an empty area of the flow diagram and deselect **`Show Columns`** to turn the option off. You can also click the tables included in the query and set an alias in the properties pane for each. These aliases help simplify the SQL code generated in the query. The **`Designer`** tab is shown in the following display.

Display 17.9 Sample Designer Tab



CAPITAL_CITY_DATA Properties

Name	Value
Name	CAPITAL_CITY...
Create View	No
Number of Ro...	
Bulkload to DB...	No
Bulkload Options	
Additional Opti...	
Description	
Checked Out By	ETL Dev 4
Table Name	CAPITAL_CITY...
Library	Target Table Li...

```
proc sql;
   create table tgtlib."CAPITAL_CITY_DATA"n as
      select
         "UNITEDSTATES"n."Capital"n length = 35
                                    format = $35.
                                    informat = $35.
                                    label = 'Capital',
         "UNITEDSTATES"n."Area"n length = 8
                                 format = Best.,
         "" as "State"n length = 2
                        format = $2.
                        informat = $2.,
         "" as "Code"n length = 2
                       format = $2.,
         "USCITYCOORDS"n."Latitude"n length = 8,
         "USCITYCOORDS"n."Longitude"n length = 8,
         "" as "Region"n length = 8
                         label = 'Region code',
```

Note that the query is shown in the Navigate pane, complete with the aliases that were set for the source tables. The process flow for the query is displayed on the **Create** tab. Finally, you can review the code for the query in the SQL Join properties pane. You can see the SQL code for the query on the **Source** tab.

Additional Information

For detailed information about specific usage issues with the SQL Join transformation, see the "SAS Data Integration Studio Usage Notes" topic in SAS Data Integration Studio help.

Configuring a SELECT Clause

Problem

You want to configure the SELECT clause for an SQL query that is used in a SAS Data Integration Studio job. This clause defines which columns will be read from the source tables and which columns will be saved in the query result tables. You must review the automappings for the query, and you might need to create one or more derived expressions for the query.

Solution

You need to use the **Select** tab on the **Designer** tab of the properties window for the SQL Join transformation.

Tasks

Configure the SELECT Clause with the Select Tab

Perform the following steps to configure the SELECT clause for the SQL query:

1 Click **Select** in the Navigate pane to access the **Select** tab.

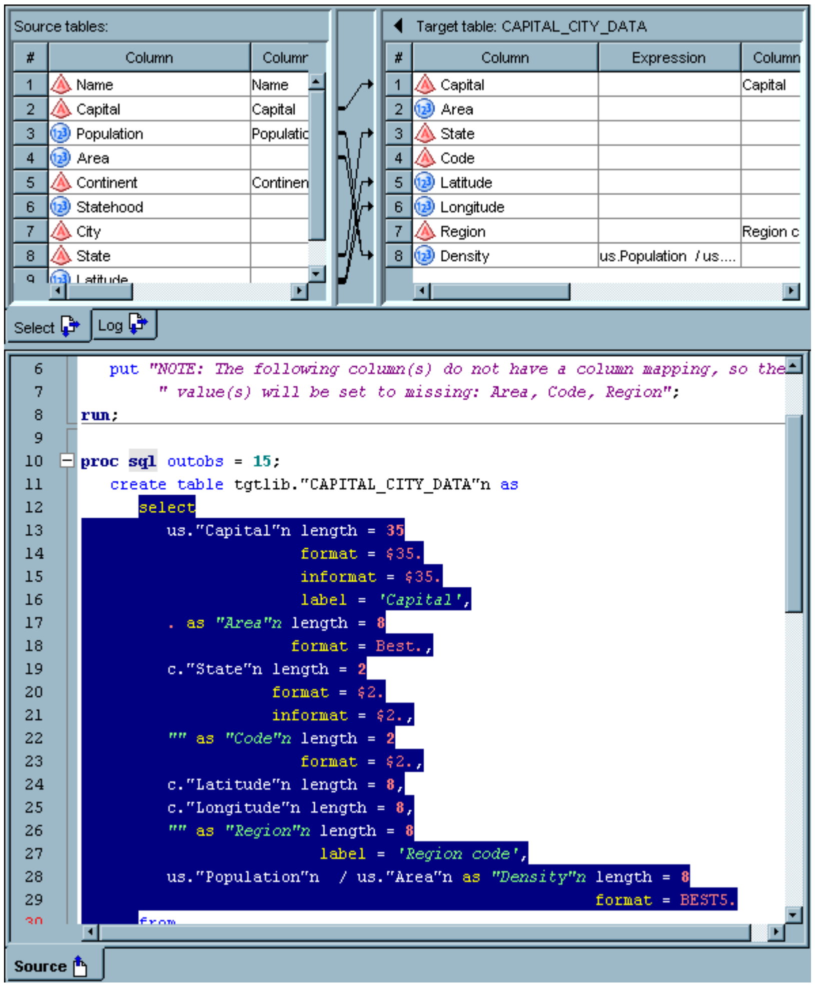
2 Review the automappings to ensure that the columns in the source table are mapped to corresponding tables in the target table. If some columns are not mapped, right-click in an empty area of the **Select** tab and click **Quick Map** in the pop-up menu.

3 Perform the following steps if you need to create a derived expression for a column in the target table for the sample query:

  - Click the drop-down menu in the **Expression** field, and click **Advanced**. The Expression Builder window displays.
  - Enter the expression that you need to create into the **Expression Text** field. (You can use the **Data Sources** tab to drill down to the column names.) For information about the Expression Builder window, see the "Expression Builder" topic in SAS Data Integration Studio help. Click **OK** to close the window.
  - Click **Yes** when the Warning window prompts you to update your mappings to take account of the expression that you just created.
  - Review the data in the row that contains the derived expression. Ensure that the column formats are appropriate for the data that will be generated by the expression. Change the formats as necessary.

4 Click **Apply** to save the SELECT clause settings to the query. The following display depicts a sample **Select** tab.

Display 17.10 Sample Select Tab Settings



Source tables:

#	Column	Columr
1	Name	Name
2	Capital	Capital
3	Population	Populatic
4	Area	
5	Continent	Continen
6	Statehood	
7	City	
8	State	
9	Latitude	

Target table: CAPITAL_CITY_DATA

#	Column	Expression	Column
1	Capital		Capital
2	Area		
3	State		
4	Code		
5	Latitude		
6	Longitude		
7	Region		Region c
8	Density	us.Population / us....	

```
    put "NOTE: The following column(s) do not have a column mapping, so the
         " value(s) will be set to missing: Area, Code, Region";
run;

proc sql outobs = 15;
   create table tgtlib."CAPITAL_CITY_DATA"n as
      select
         us."Capital"n length = 35
                       format = $35.
                       informat = $35.
                       label = 'Capital',
         . as "Area"n length = 8
                      format = Best.,
         c."State"n length = 2
                    format = $2.
                    informat = $2.,
         "" as "Code"n length = 2
                       format = $2.,
         c."Latitude"n length = 8,
         c."Longitude"n length = 8,
         "" as "Region"n length = 8
                         label = 'Region code',
         us."Population"n  / us."Area"n as "Density"n length = 8
                                                      format = BEST5.
      from
```

Note that the SQL code for the SELECT clause is displayed on the **Source** tab.

Adding a CASE Expression

Problem

You want to create a CASE expression to incorporate conditional processing into an SQL query contained in a SAS Data Integration Studio job. The CASE expression can be added to the following parts of a query:

- □ a SELECT statement
- □ a WHERE condition
- □ a HAVING condition
- □ a JOIN condition

Solution

You can use the CASE Expression window to add a conditional expression to the query.

Tasks

Add a CASE Expression to an SQL Query in the Designer Tab

Perform the following steps to add a CASE expression to the SQL query in the Designer tab of the properties window for the SQL Join transformation:

1 Access the CASE Expression window. To do this, click **CASE** in the drop-down menu for an Operand in a WHERE, HAVING, or JOIN condition. You can also access the **CASE** option in the Expression column for any column listed in the **Target table** field in the **Select** tab.

2 Click **New** to begin the first condition of the expression. An editable row appears in the table.

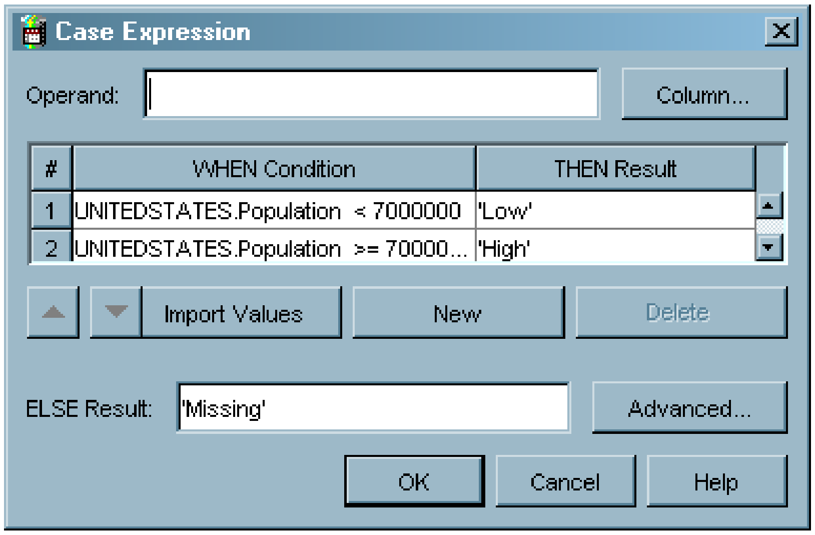
3 Enter the appropriate WHEN condition and THEN result for the first WHEN/THEN clause. For information about entering these values, see the "Case Expression Window" topic in SAS Data Integration Studio help.

4 Add the remaining WHEN/THEN clauses. You need to add one row for each clause.

5 Enter an appropriate value in the **ELSE Result** field. This value is returned for any row that does not satisfy one of the WHEN/THEN clauses.

6 Click **OK** to save the CASE expression and close the window. The following display depicts a sample completed CASE Expression window.

Display 17.11 Sample Completed CASE Expression Window



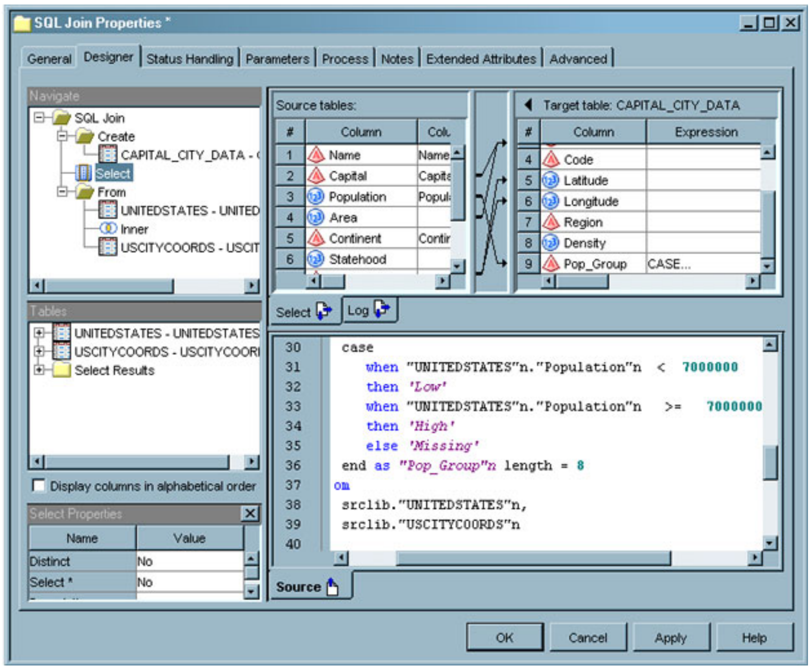
Operand:

#	WHEN Condition	THEN Result
1	UNITEDSTATES.Population < 7000000	'Low'
2	UNITEDSTATES.Population >= 70000...	'High'

ELSE Result: 'Missing'

Note that the Operand field is blank. You can specify the operand only when the conditions in the CASE expression are all equality tests. The expression in this sample query uses comparison operators. Therefore, the UNITEDSTATES.Population column name must be entered for each WHEN condition in the expression. The following display depicts the **Select** tab and the highlighted CASE expression for a sample query.

Display 17.12 Sample CASE Expression Query



Source tables:

#	Column	Colu
1	Name	Name
2	Capital	Capita
3	Population	Popul
4	Area	
5	Continent	Contir
6	Statehood	

Target table: CAPITAL_CITY_DATA

#	Column	Expression
4	Code	
5	Latitude	
6	Longitude	
7	Region	
8	Density	
9	Pop_Group	CASE...

```
case
   when "UNITEDSTATES"n."Population"n  <  7000000
   then 'Low'
   when "UNITEDSTATES"n."Population"n   >=   7000000
   then 'High'
   else 'Missing'
end as "Pop_Group"n length = 8
om
srclib."UNITEDSTATES"n,
srclib."USCITYCOORDS"n
```

Name	Value
Distinct	No
Select *	No

Note that the Population column in the **`Source tables`** field in the **`Select`** tab is mapped to both the Population and the Pop_Group columns in the **`Target table`** field. The second mapping, which links Population to Pop_Group, is created by the CASE expression described in this topic.

Creating or Configuring a WHERE Clause

Problem

You want to configure the WHERE clause for an SQL query that is used in a SAS Data Integration Studio job. The conditions included in this clause determine which subset of the data from the source tables is included in the query results that are collected in the target table.

Solution

You can use the **`Where`** tab in the **`Designer`** tab of the properties window for the SQL Join transformation.

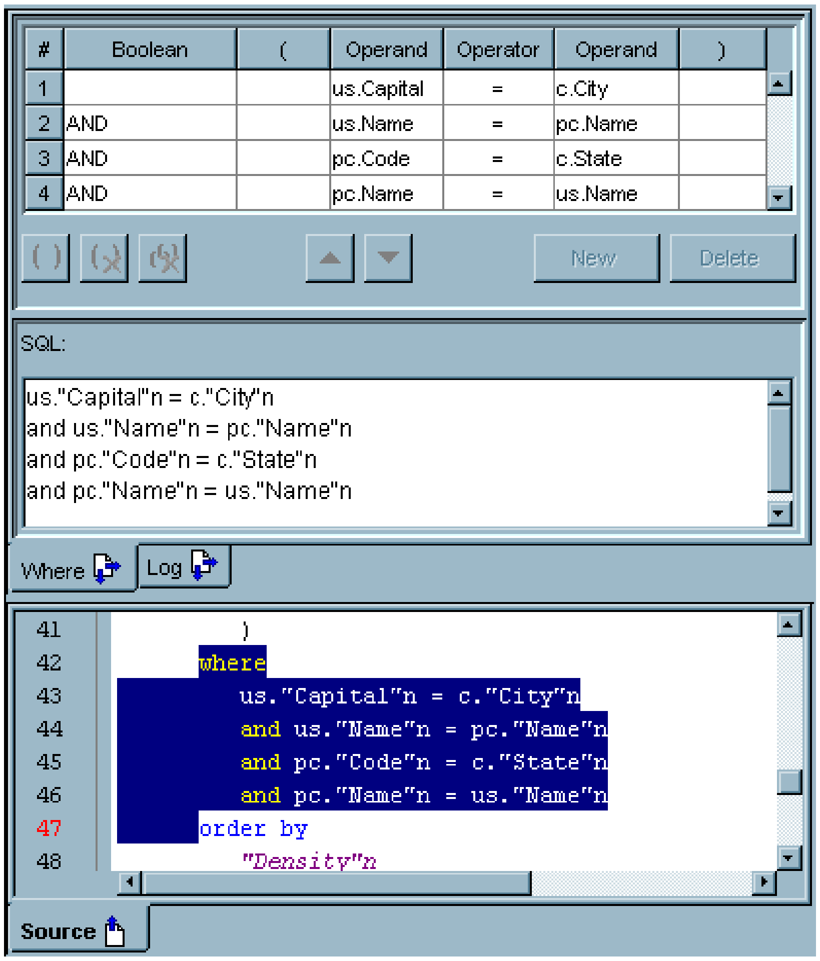
Tasks

Configure the WHERE Clause with the Where Tab

The WHERE clause for the query is an SQL expression that creates subsets of the source tables in the SQL query. It also defines the join criteria for joining the source tables and the subquery to each other by specifying which values to match. Perform the following steps to configure the **`Where`** tab:

1. If the **`Where`** clause object is missing from the process flow in the **`Create`** tab, double-click **`Where`** in the SQL Clauses pane. The **`Where`** clause object is added to the query flow in the **`Create`** tab. Note that **`Where`** clause objects are automatically populated into the **`Create`** tab. The WHERE clause is not automatically generated under the following circumstances:
   - the query contains only one source table
   - no relationship was found during the auto-join process
2. Click **`Where`** in the Navigate pane to access the **`Where`** tab.
3. Click **`New`** on the **`Where`** tab to begin the first condition of the expression. An editable row appears in the table near the top of the tab.
4. Enter the appropriate operands and operator for the first condition. For information about entering these values, see the "Where/Having/Join Tab" topic in SAS Data Integration Studio help.
5. Add the remaining conditions for the WHERE clause. You need to add one row for each condition.
6. Click **`Apply`** to save the WHERE clause settings to the query. The conditions created for the sample query are depicted in the SQL code generated in this step in the **`SQL`** field, as shown in the following display.

Display 17.13 Sample Where Tab Settings



#	Boolean	(	Operand	Operator	Operand	)
1			us.Capital	=	c.City	
2	AND		us.Name	=	pc.Name	
3	AND		pc.Code	=	c.State	
4	AND		pc.Name	=	us.Name	

SQL:

```
us."Capital"n = c."City"n
and us."Name"n = pc."Name"n
and pc."Code"n = c."State"n
and pc."Name"n = us."Name"n
```

```
               )
         where
            us."Capital"n = c."City"n
            and us."Name"n = pc."Name"n
            and pc."Code"n = c."State"n
            and pc."Name"n = us."Name"n
         order by
            "Density"n
```

Note that the SQL code for the WHERE clause that is shown in the **SQL** field is identical to the highlighted WHERE clause code displayed in the **Source** tab.

Adding a GROUP BY Clause and a HAVING Clause

Problem

You want to group your results by a selected variable. Then, you want to subset the number of groups displayed in the results

Solution

You can add a GROUP BY clause to group the results of your query. You can also add a HAVING clause that uses an aggregate expression to subset the groups returned by the GROUP BY clause that are displayed in the query results.

Tasks

Add a GROUP BY Clause to an SQL Query in the Designer Tab

Perform the following steps to add a GROUP BY clause to the SQL query in the **Designer** tab of the properties window for the SQL Join transformation:

1. Click **Create** in the Navigate pane to gain access to the **Create** tab and the SQL Clauses pane.
2. Double-click **Group by** in the SQL Clauses pane. The **Group by** object is added to the query flow in the **Create** tab.
3. Click **Group by** in the Navigate pane to access the **Group by** tab.
4. Select the column that you want to use for grouping the query results from the **Available columns** field. Then, move the column to the **Group by columns** field.
5. Click **Apply** to save the GROUP BY clause settings to the query. The following display depicts a sample SQL query grouped with a GROUP BY clause.

Display 17.14 Sample SQL Query Grouped With a GROUP BY Clause

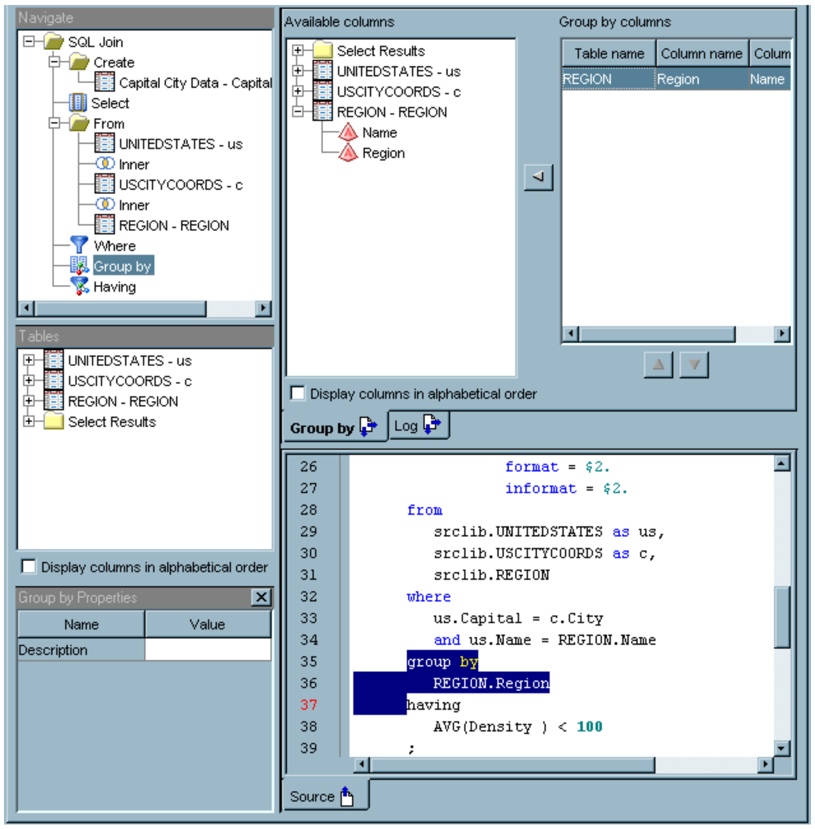


Table name	Column name	Colum
REGION	Region	Name

```
                format = $2.
                informat = $2.
        from
            srclib.UNITEDSTATES as us,
            srclib.USCITYCOORDS as c,
            srclib.REGION
        where
            us.Capital = c.City
            and us.Name = REGION.Name
        group by
            REGION.Region
        having
            AVG(Density ) < 100
        ;
```

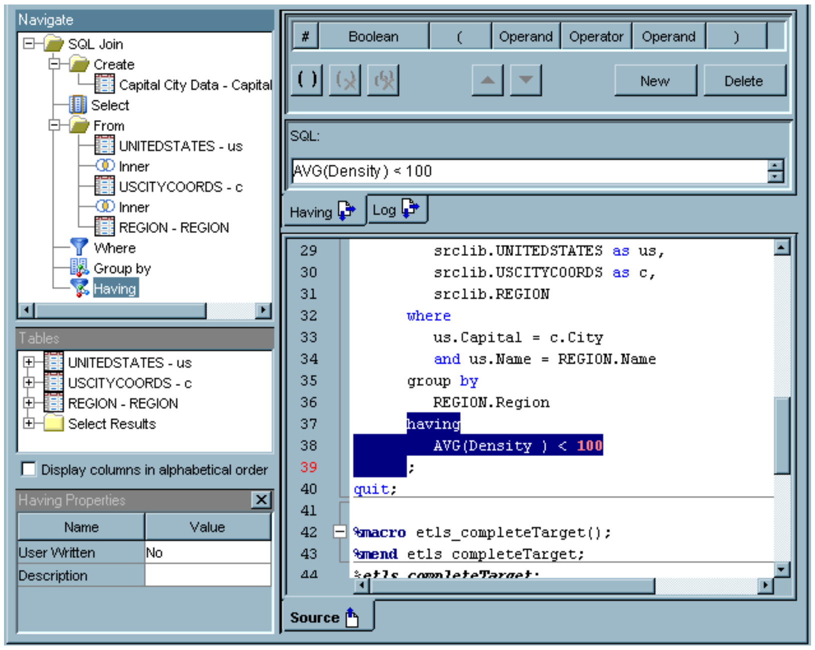
Note that the group by column is set on the **Group by** tab, and the resulting SQL code is highlighted on the **Source** tab. The GROUP BY clause in the sample query groups the results of the query by the region of the United States.

Add a HAVING Clause to an SQL Query on the Designer Tab

Perform the following steps to add a **`Having`** clause to the SQL query on the **`Designer`** tab of the properties window for the SQL Join transformation:

1. Click **`Create`** in the Navigate pane to gain access to the **`Create`** tab and the SQL Clauses pane.
2. Double-click **`Having`** in the SQL Clauses pane. The **`Having`** object is added to the query flow on the **`Create`** tab.
3. Click **`Having`** in the Navigate pane to access the **`Having`** tab.
4. Click **`New`** on the **`Having`** tab to begin the first condition of the expression. An editable row appears in the table near the top of the tab.
5. Enter the appropriate operands and operator for the first condition. For information about entering these values, see the "Where/Having/Join Tab" topic in SAS Data Integration Studio help.
6. Add the remaining conditions for the HAVING clause. You need to add one row for each condition.
7. Click **`Apply`** to save the HAVING clause settings to the query. The condition created for the sample query is depicted in the SQL code generated in this step in the **`SQL`** field, as shown in the following display.

Display 17.15 Sample SQL Query Subsetted with a HAVING Clause



Display columns in alphabetical order

Having Properties

Name	Value
User Written	No
Description	

SQL:

```
AVG(Density) < 100
```

```
         srclib.UNITEDSTATES as us,
         srclib.USCITYCOORDS as c,
         srclib.REGION
      where
         us.Capital = c.City
         and us.Name = REGION.Name
      group by
         REGION.Region
      having
         AVG(Density ) < 100
      ;
quit;

%macro etls_completeTarget();
%mend etls completeTarget;
%etls completeTarget;
```

Note that the SQL code for the HAVING clause that is shown in the **`SQL`** field is identical to the highlighted HAVING clause code displayed on the **`Source`** tab. The HAVING clause subsets the groups that are included in the results for the query. In the sample, only the regions with an average population density of less than 100 are included in the query results. Therefore, the Western and Mountain region results are included, but the Eastern and Midwestern region results are not.

Adding an ORDER BY Clause

Problem

You want to sort the output data in an SQL query that is included in a SAS Data Integration Studio job.

Solution

You can use the **`Order by`** tab on the **`Designer`** tab in the properties window for the SQL Join transformation.

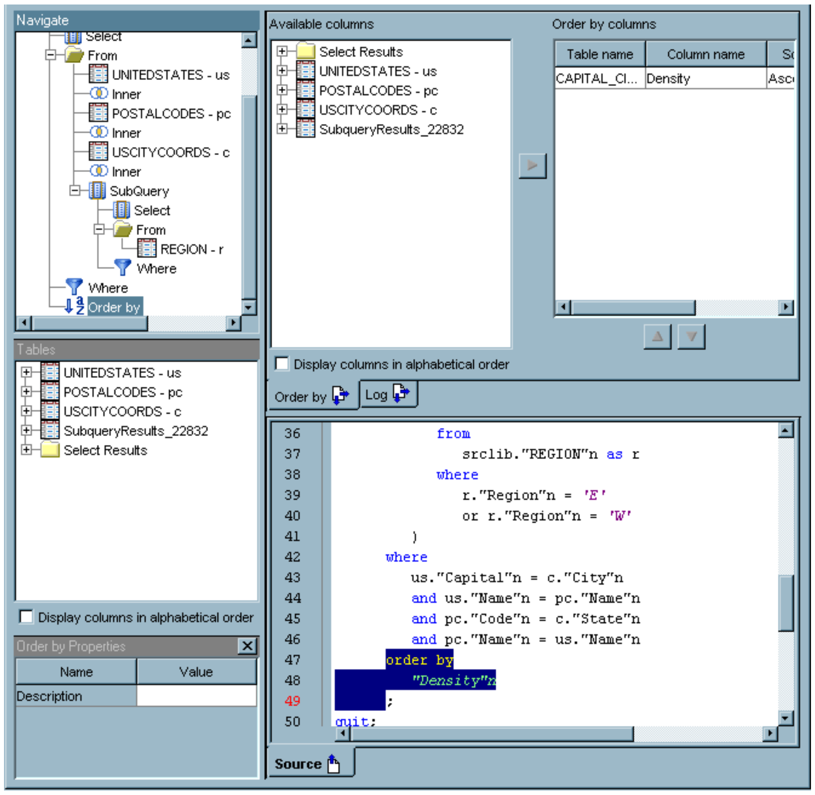
Tasks

Add an ORDER BY Clause to an SQL Query in the Designer Tab

You can add an ORDER BY clause to establish a sort order for the query results. Perform the following steps to add an ORDER BY clause to the SQL query in the **`Designer`** tab of the properties window for the SQL Join transformation:

1. Click **`Create`** in the Navigate pane to gain access to the **`Create`** tab and the SQL Clauses pane.
2. Double-click **`Order by`** in the SQL Clauses pane. The **`Order by`** object is added to the query flow in the **`Create`** tab.
3. Right-click the **`Order by`** object and click **`Edit`** in the pop-up menu.
4. Select the column that you want to use for ordering the query results from the **`Available columns`** field. Then, move the column to the **`Order by columns`** field. Finally, enter a value in the **`Sort Order`** field to determine whether the results are sorted in ascending or descending order.
5. Click **`Apply`** to save the ORDER BY clause settings to the query. The following display depicts a sample SQL query with an ORDER BY clause.

Display 17.16 Sample SQL Query Sorted With an ORDER BY Clause



```
            from
               srclib."REGION"n as r
            where
               r."Region"n = 'E'
               or r."Region"n = 'W'
         )
      where
         us."Capital"n = c."City"n
         and us."Name"n = pc."Name"n
         and pc."Code"n = c."State"n
         and pc."Name"n = us."Name"n
      order by
         "Density"n
      ;
quit;
```

Note that Order by columns is set in the **Order by** tab, and the resulting SQL code is highlighted in the **Source** tab.

Adding Subqueries

Problem

You want to add one or more subqueries to an existing SQL query by using the **Designer** tab of the properties window for the SQL Join transformation.

Solution

Use the **Subquery** object in the **Designer** tab that is located in the properties window for the SQL Join transformation. The sample job used in "Add a Subquery to an Input Table" on page 312 adds a subquery to an input table. This subquery reduces the amount of data that is processed in the main SQL query because it runs and subsets data before the SELECT clause is run. "Add a Subquery to an SQL Clause" on page 314 covers adding a subquery to a SELECT, WHERE, or HAVING clause in an SQL query.

Tasks

Add a Subquery to an Input Table

You can add the source and target tables directly to the process flow diagram for the job. You can also add a table, join, or subquery to a job by dragging and dropping it on an empty source or target drop zone in the **Create** tab in the **Designer** tab of the properties window for the SQL Join transformation. If you drop a table on an existing table in the **Designer** tab, the new table will replace the existing table.

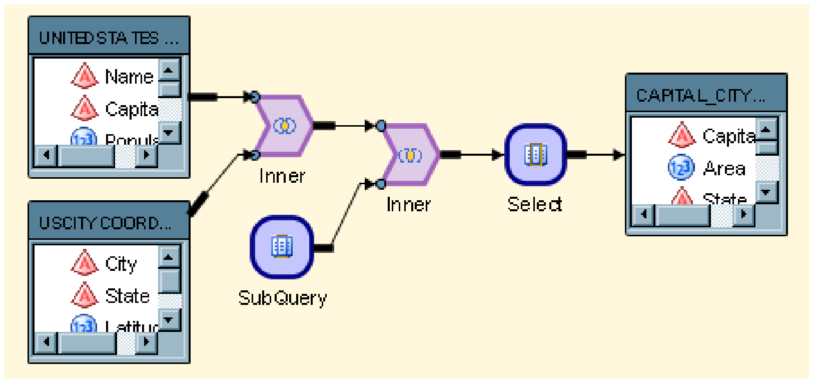
You can even add a new drop zone to the query flow in the **Create** tab. To perform this task, select one of the join icons from the Joins directory in the SQL Clauses pane and drop it on the Select object, a table, or an existing join in the flow. The join and its drop zone will be displayed in the query flow. Use this method to add a subquery to the job.

Perform the following steps to create a subquery that refines the SQL query:

1 Select **Inner** in the Joins folder in the SQL Clauses pane, and drop it on the Select object in a query flow displayed on the **Create** tab. An Inner object is added to the query flow.

2 Select **SubQuery** in the Select Clauses folder in the SQL Clauses pane. Drop it in the drop zone attached to the new **Inner** object in the query flow. The subquery is added to the join, as shown in the sample job depicted in the following display.

Display 17.17 SQL Query with an Added Subquery



3 Click the **SubQuery** object. Note that the SubQuery Properties pane displays. Enter an appropriate value in the **Alias** field. (**RegionQry** was entered in the sample job.) Enter an alias here; otherwise, the subquery will fail. The system-generated name for the subquery results table is too ambiguous to be recognized as an input to the full SQL query.

4 Right-click **SubQuery**, and click **Edit**. The SubQuery pane is displayed.

5 Drop the source table on the drop zone for the **Select** clause object.

6 Enter an alias for the source table into its properties pane. This step is optional, but it does simplify the SQL code.

7 Right-click **Select** and click **Edit** to display the **Select** tab. Make sure that the source table columns are mapped properly to the RegionQry target table. Also, ensure that the **Select *** property in the Select Properties pane is set to **No**.

8 Click **SubQuery** in the Navigate pane to return to the **SubQuery** tab. Then, select **Where** in the **SQL Clauses** folder of the SQL Clause pane. Finally, drop the **Where**

icon into an empty spot in the **SubQuery** tab. A **Where** clause object is added to the **SubQuery** tab.

9 Right-click **Where** and click **Edit** to display the **Where** tab.

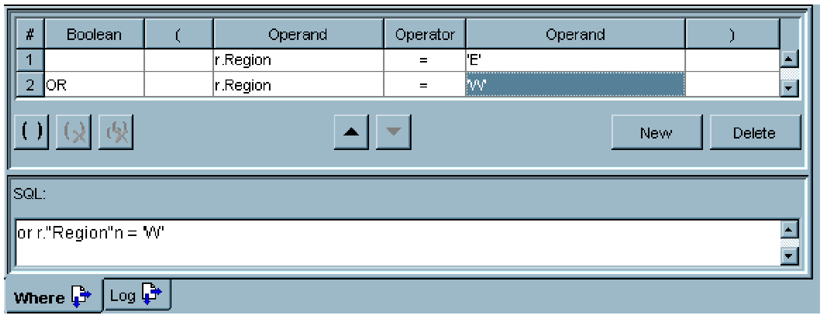
10 Click **New** on the **Where** tab to begin the first part of the expression. An editable row appears in the table near the top of the tab.

11 Create your first WHERE condition. In this example, a subset of the **Region** column from the Region table to select values from the eastern region was created. To recreate the condition, click the drop-down menu in the **Operand** field on the left side of the row, and click **Choose column(s)**. Then, drill down into the Region table, and select the **Region** column. The value **r.Region** displays in the field.

12 Keep the defaulted value of **=** in the **Operator** field. Enter the value **'E'** in the **Operand** field on the right side of the row.

13 Create the remaining conditions for the WHERE statement. Review the SQL code generated in this step in the SQL field, as shown in the following display.

Display 17.18 Where Tab in the Subquery



#	Boolean	(	Operand	Operator	Operand	)
1			r.Region	=	'E'	
2	OR		r.Region	=	'W'	

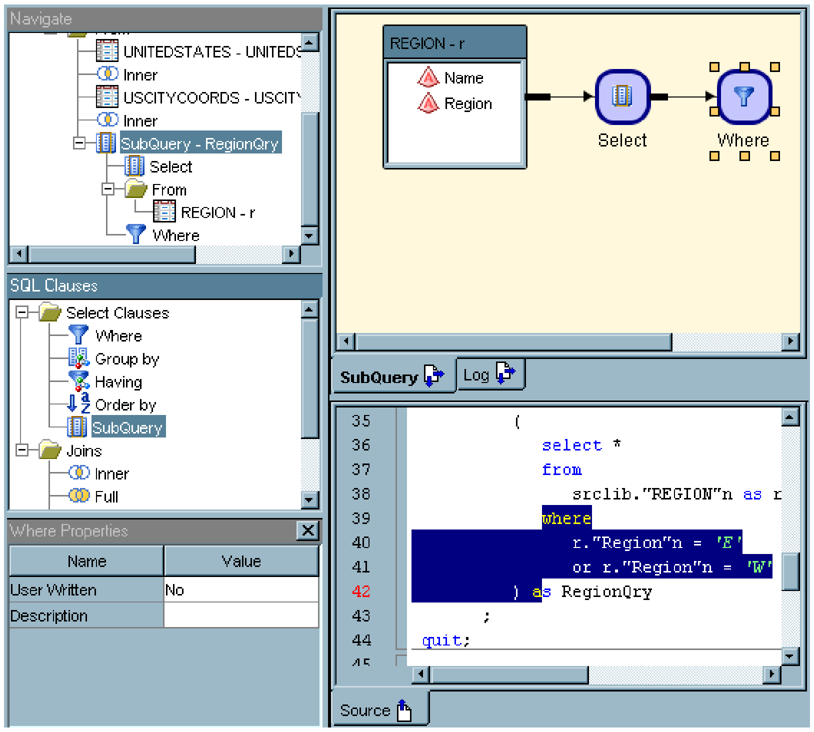
```
or r."Region"n = 'W'
```

14 A connection is required between the source table for the subquery and the target table for the query. To recreate the sample, right-click in the **Target table** field of the **Select** tab and click **New Column** in the pop-up menu.

15 Enter name of the subquery source table in the **Name** field. Then, make sure that the new column has the appropriate data type.

16 Click **Apply** to save the new column into the target table. You will have to add a mapping for the subquery to the main query SELECT clause, and add the subquery to the main query WHERE clause. The following display depicts the input table subquery.

Display 17.19 Sample Input Table Subquery



Name	Value
User Written	No
Description	

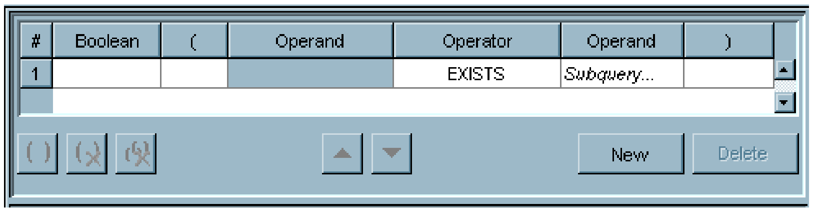
```
            (
                select *
                from
                    srclib."REGION"n as r
                where
                    r."Region"n = 'E'
                    or r."Region"n = 'W'
            ) as RegionQry
        ;
    quit;
```

You can compare the tree view of the subquery in the Navigate pane to the process flow in the **SubQuery** tab and the highlighted code in the **Source** tab. Note that you can add a subquery any place that you can add a table or a column.

Add a Subquery to an SQL Clause

You can also add a subquery to SELECT, WHERE, HAVING clauses in SQL queries. The following display shows how a subquery can be added as a condition to a WHERE clause.

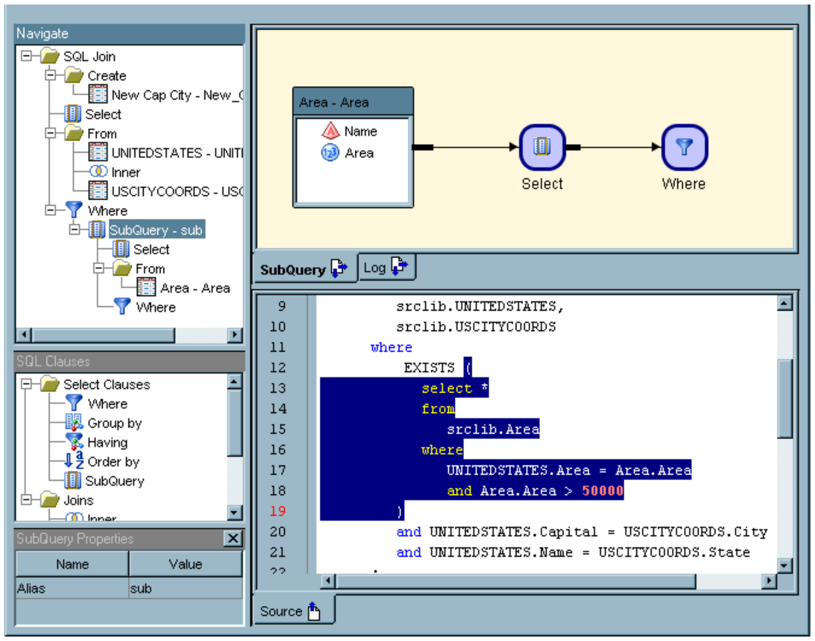
Display 17.20 Add a Subquery to a WHERE Clause



#	Boolean	(	Operand	Operator	Operand	)
1				EXISTS	*Subquery...*	

Note that the subquery is connected to the WHERE clause with the EXISTS operator, which you can select from the drop-down menu in the **Operator** field. To add the subquery, click in the **Operand** field on the right-hand side of the **Where** tab. Then, click **Subquery** from the drop-down menu. The following display shows the completed sample subquery.

Display 17.21 Sample WHERE Clause Subquery



Name	Value
Alias	sub

```
        srclib.UNITEDSTATES,
        srclib.USCITYCOORDS
    where
         EXISTS (
           select *
           from
              srclib.Area
           where
              UNITEDSTATES.Area = Area.Area
              and Area.Area > 50000
        )
        and UNITEDSTATES.Capital = USCITYCOORDS.City
        and UNITEDSTATES.Name = USCITYCOORDS.State
```

The subquery includes a SELECT clause, a WHERE clause, and a HAVING clause. You can compare the tree view of the subquery in the Navigate pane to the process flow in the **SubQuery** tab and the highlighted code in the **Source** tab. The HAVING clause nested into the WHERE clause enables you to further refine data selection in the query.

Submitting an SQL Query

Problem

You want to submit an SQL query, either to verify that it will work properly or to run it as part of a SAS Data Integration Studio job.

Solution

You can submit an SQL query in two distinct contexts. First, you can submit it from the **Designer** tab of the SQL Join transformation. This approach can be helpful when you want to make sure that your query runs properly and returns the data that you're seeking. You can also submit the query when you run the SAS Data Integration Studio job that contains the SQL Join transformation.

Tasks

Submit a Query from the Designer Tab of the SQL Join Transformation

Perform the following steps to submit a query from the **Designer** tab:

1 Submit the query in one of the following ways:
   - □ Click **Submit** on the SAS Data Integration Studio menu bar.
   - □ Right-click on the **Create**, **SubQuery**, **Where**, **Having**, **Join**, or **Source** tab. Then, click **Submit**.
   - □ Click **Submit** on the SAS Data Integration Studio **SQL** menu.

   If **Run Prior Steps on Submit** is selected on the pop-up menu or the **SQL** menu, the steps that are placed before the SQL query code are submitted. (These steps are used to create the source tables for the query.) When deselected, this option runs the SQL query code only. This setting enables you to test changes to the SQL query. This option takes effect only when work tables are used as the source.

2 Validate the query as needed. For example, you can check the properties of the target table. You can also review the data populated into the target table in the View Data window. Finally, you can examine the **Log** tab to verify that the query was submitted successfully or to troubleshoot an unsuccessful submission.

Note: You can use the Job Status Manager in SAS Data Integration Studio to cancel the SQL query. The SQL Join transformation is displayed as a row in the Job Status Manager. You can right-click the row and click **Cancel Job** to cancel the query. The SQL Join transformation is currently the only transformation that supports this type of cancellation. △

Submit a Query as a Part of a SAS Data Integration Studio Job

Perform the following steps to submit a query from the SAS Data Integration Studio job:

1 Submit the query in one of the following ways:
   - □ Click **Submit** on the SAS Data Integration Studio menu bar.
   - □ Right-click in the Process Designer window. Then, click **Submit**.
   - □ Click **Submit** on the SAS Data Integration Studio Process menu.

2 Validate the job as needed. For example, you can check the properties of the target table. You can also review the data populated into the target table in the View Data window. Finally, you can examine the **Log** tab to verify that the job was submitted successfully or to troubleshoot an unsuccessful submission.

Joining a Table to Itself

Problem

You want to join a table to itself.

Solution

You can join the table to itself by creating the second version of the table with an alias. Then, you can take some of the variables used in a query from the original table and take the remaining variables from the newly created copy of the table.

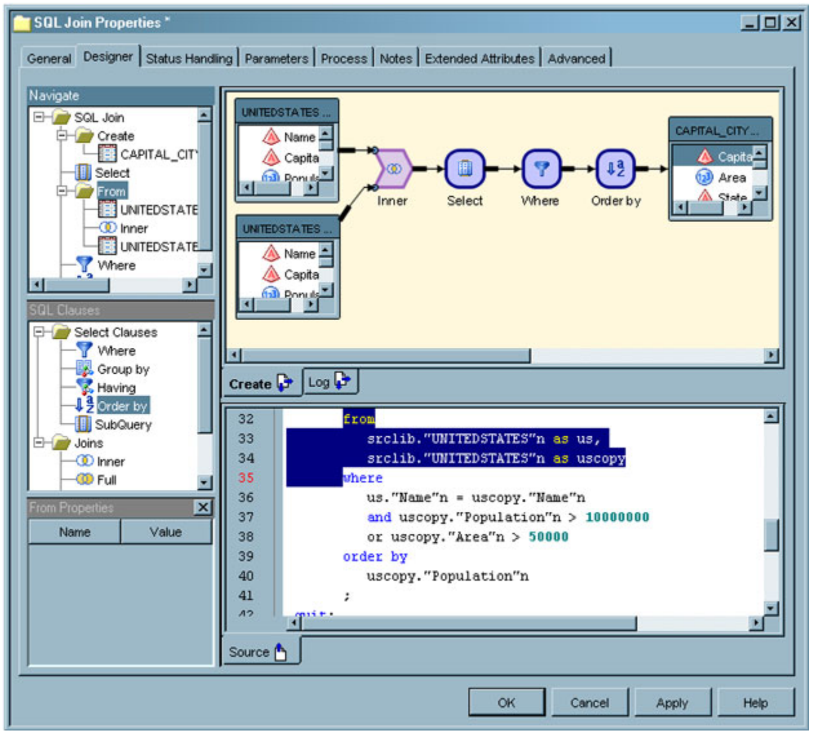
Tasks

Join the Table to Itself

Perform the following steps to join a table to itself and use the resulting hierarchy of tables in a query:

1. Create an SQL query in an empty job. The query should contain the SQL Join transformation, at least one source table, and a target table.
2. Open the **`Designer`** tab in the SQL Join transformation. Click **`Create`** in the Navigate pane to access the **`Create`** tab and the SQL Clauses pane.
3. Drop a join from the SQL Clauses pane on the **`Select`** object on the **`Create`** tab.
4. Drop the table used as a source table for the query on the drop zone for the join. You will be prompted to supply an alias for the table because it is already being used as a source table for the query.
5. Enter the alias in the **`Alias`** field of the properties pane for the table.
6. Click **`Apply`** to save the alias for the table that you just added.
7. Complete any additional configuration needed to finish the query. The following display shows a sample job that includes a table joined to itself.

Display 17.22 Sample Job with a Table Joined to Itself



```
        from
           srclib."UNITEDSTATES"n as us,
           srclib."UNITEDSTATES"n as uscopy
        where
           us."Name"n = uscopy."Name"n
           and uscopy."Population"n > 10000000
           or uscopy."Area"n > 50000
        order by
           uscopy."Population"n
        ;
```

Note that the table jobs shown on the **Create** tab are reflected in the FROM clause highlighted on the **Source** tab. The query shown in the sample job pulls the Name variable from the original table (denoted with the **us** alias). However, it pulls the Population and Area variables from the copy of the original table (denoted with the **uscopy** alias).

Using Parameters with an SQL Join

Problem

You want to include an SQL Join transformation in a parameterized job. Then, the parameterized job is run in an iterative job. The iterative job contains a control loop in which one or more processes are executed multiple times. This arrangement allows you to iteratively run a series of tables in a library through your SQL query. For example, you could process a series of 50 tables that represent each of the 50 states in the United States through the same SQL query.

Solution

You can create one or more parameters on the **Parameters** tab in the properties window for the SQL Join transformation. Then, you can use the parameters to tie the SQL Join transformation to the other parts of the parameterized job and the iterative job that contains it. The following prerequisites must be satisfied before the SQL Join transformation can work in this iterative setting:

- The SQL Join transformation must be placed in a parameterized job. See "Creating a Parameterized Job" on page 335.
- One or more parameters must be set for the input and output tables for the parameterized job. See "Set Input and Output Parameters" on page 336.
- One or more parameters must be set for the parameterized job. See "Set Parameters for the Job" on page 337.
- The parameterized job must be embedded in an iterative job. See "About Iterative Jobs" on page 331.
- The parameters from the parameterized job must be mapped on the **Parameter Mapping** tab of the properties window for the iterative job. See "About Iterative Jobs" on page 331.
- The tables that you need to process through query created in the SQL Join transformation must be included in the control table for the iterative job. See "Creating a Control Table" on page 338.

Constructing a SAS Scalable Performance Data Server Star Join

Problem

You want to construct SAS Scalable Performance Data (SPD) Server star joins.

Solution

You can use the SAS Data Integration Studio SQL Join transformation to construct SAS SPD Server star joins when you use SAS SPD Server version 4.2 or later.

Tasks

Construct an SPD Server Star Join

Star joins are useful when you query information from dimensional models that are constructed of two or more dimension tables that surround a centralized fact table referred to as a star schema. SAS SPD Server star joins are queries that validate, optimize, and execute SQL queries in the SAS SPD Server database for performance. If the star join is not used, the SQL will be processed in SAS SPD Server using pair-wise joins, which require one step for each table to complete the join. When the SAS SPD Server options are set, the star join is enabled.

You must meet the following requirements to enable a star join SAS SPD Server:

- □ All dimension tables must surround a single fact table.
- □ Dimension-to-fact table joins must be equal joins, and there should be one join per dimension table.
- □ You must have two or more dimension tables in the join condition.
- □ The fact table must have at least one subsetting condition placed on it.
- □ All subsetting and join conditions must be specified in the WHERE clause.
- □ Star join optimization must be enabled through the setting of options on the SAS SPD Server library.

In order to enable star join optimization, code that will run on the generated Pass SAS SPD Server system library must have the following options added to the library:

- □ **LIBGEN=YES** *
- □ **IP=YES**

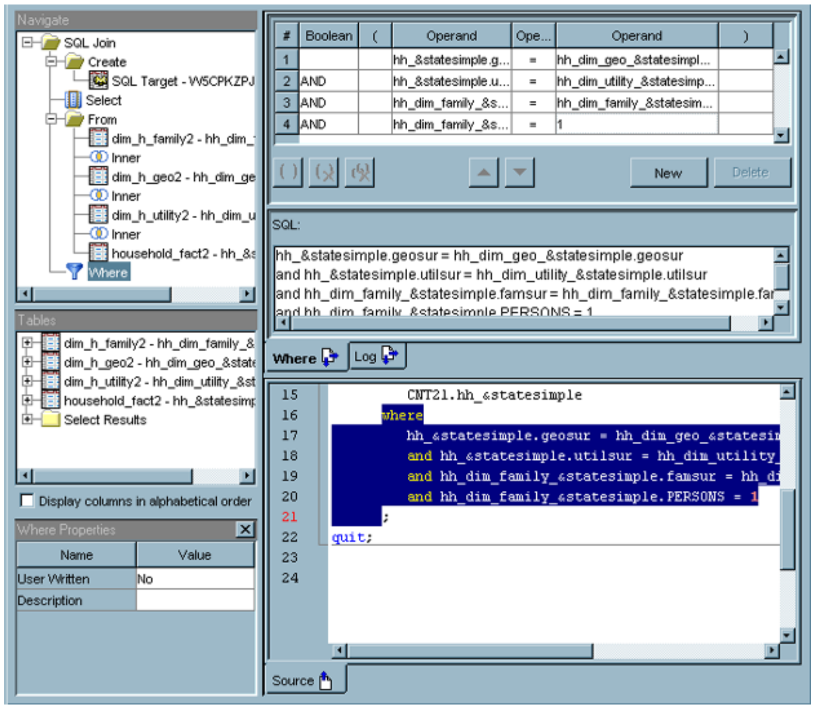
Here is a commented example of a WHERE clause that will enable a SAS SPD Server star join optimization:

```
where
/* dimension1 equi-joined on the fact */
   hh_&statesimple.geosur = hh_dim_geo_&statesimple.geosur
/* dimension2 equi-joined on the fact */
   and hh_&statesimple.utilsur = hh_dim_utility_&statesimple.utilsur
/* dimension3 equi-joined on the fact */
   and hh_dim_family_&statesimple.famsur = hh_dim_family_&statesimple.famsur
/* subsetting condition on the fact */
```

```
      and hh_dim_family_&statesimple.PERSONS = 1
   ;
```

The following display depicts the **Designer** tab in an SQL Join transformation that is included in a job that constructs an SPD Server star join.

Display 17.23 Sample SPD Server Star Join Code in an SQL Query



#	Boolean	(	Operand	Ope...	Operand	)
1			hh_&statesimple.g...	=	hh_dim_geo_&statesimpl...	
2	AND		hh_&statesimple.u...	=	hh_dim_utility_&statesimp...	
3	AND		hh_dim_family_&s...	=	hh_dim_family_&statesim...	
4	AND		hh_dim_family_&s...	=	1	

```
hh_&statesimple.geosur = hh_dim_geo_&statesimple.geosur
and hh_&statesimple.utilsur = hh_dim_utility_&statesimple.utilsur
and hh_dim_family_&statesimple.famsur = hh_dim_family_&statesimple.far
and hh_dim_family_&statesimple.PERSONS = 1
```

```
            CNT21.hh_&statesimple
        where
            hh_&statesimple.geosur = hh_dim_geo_&statesi
            and hh_&statesimple.utilsur = hh_dim_utility_
            and hh_dim_family_&statesimple.famsur = hh_di
            and hh_dim_family_&statesimple.PERSONS = 1
        ;
quit;
```

Name	Value
User Written	No
Description	

Note that the SAS SPD Server requires all subsetting to be implemented on the **Where** tab in the SQL Join transformation. For more information about SAS SPD Server support for star joins, see the "SAS Scalable Performance Data Server 4.4: User's Guide." When the code is properly configured, the following output is generated in the log: **SPDS_NOTE: STARJOIN optimization used in SQL execution**

Additional Information

For detailed information about specific usage issues with the SQL Join transformation, see the "SAS Data Integration Studio Usage Notes" topic in SAS Data Integration Studio help.

Optimizing SQL Processing Performance

Problem

Joins are a common and resource-intensive part of SAS Data Integration Studio. SAS SQL implements several well-known join algorithms: sort-merge, index, and hash.

You can use common techniques to aid join performance, irrespective of the algorithm chosen. Conditions often cause the SAS SQL optimizer to choose the sort-merge algorithm; techniques that improve sort performance also improve sort-merge join performance. However, understanding and leveraging index and hash joins will enhance performance.

It is common in SAS Data Integration Studio to perform lookups between tables. Based on key values in one table, you look up matching keys in a second table and retrieve associated data in the second table. SQL joins can perform lookups. However, SAS and SAS Data Integration Studio provide special lookup mechanisms that typically outperform a join. The problems associated with joins are similar to those with sorting:

- Join performance seems slow.
- You have trouble influencing the join algorithm that SAS SQL chooses.
- You experience higher than expected disk space consumption.
- You have trouble operating SAS SQL joins with RDBMS data.

Solution

Review the techniques explained in the following topics:

- "Debugging an SQL Query" on page 296
- "Enabling Pass-Through Processing" on page 325
- "Influencing the Join Algorithm" on page 322
- "Performing General Data Optimization" on page 321
- "Understanding Automatic Joins" on page 289
- "Setting the Implicit Property for a Join" on page 324
- "Selecting the Join Type" on page 292
- "Using Property Sheet Options to Optimize SQL Processing Performance" on page 327

Performing General Data Optimization

Problem

You want to streamline the data as much as possible before you run it through SQL processing in a SAS Data Integration Studio job.

Solution

You can minimize the input and output overhead for the data. You can also pre-sort the data.

Tasks

Minimize Input/Output (I/O) Processing

To help minimize I/O and improve performance, you can drop unneeded columns, minimize column widths (especially from Database Management System [DBMS] tables that have wide columns), and delay the inflation of column widths until the end of your SAS Data Integration Studio flow. (Column width inflation becomes an issue when you combine multiple columns into a single column to use a key value).

Pre-Sort Data

Pre-sorting can be the most effective means to improve overall join performance. A table that participates in multiple joins on the same join key usually benefits from pre-sorting. For example, if the ACCOUNT table participates in four joins on ACCOUNT_ID, then pre-sorting the ACCOUNT table on ACCOUNT_ID helps optimize three joins. However, the overhead associated with sorting can degrade performance. You can sometimes achieve better performance when you subset by using the list of columns in the SELECT statement and the conditions set in the WHERE clause.

Note: Integrity constraints are automatically generated when the query target to the SQL transformation is a physical table. You can control the generation of these constraints by using a Table Loader transformation between the SQL Join transformation and its physical table. △

Influencing the Join Algorithm

Problem

You want to influence the SAS SQL optimizer to choose the join algorithm that will yield the best possible performance for the SQL processing included in a SAS Data Integration Studio job. SAS SQL implements several well-known join algorithms: sort-merge, index, and hash.

Solution

There are common techniques to aid join performance, irrespective of the algorithm chosen. These techniques use options that are found on the SQL Properties pane and the properties panes for the tables found in SAS queries. However, selecting a join algorithm is important enough to merit a dedicated topic. You can use the **`Debug`** property on the SQL Join Properties pane to run the **`_method`**, which will add a trace that will indicate which algorithm is used when in the **`Log`** tab.

Tasks

Sort-Merge Joins

Conditions often cause the SAS SQL optimizer to choose the sort-merge algorithm, and techniques that improve sort performance also improve sort-merge join performance. However, understanding and using index and hash joins can provide performance gains. Sort-merge is the algorithm most often selected by the SQL optimizer. When index nested loop and hash join are eliminated as choices, a sort-merge join or simple nested loop join is used. A sort-merge sorts one table, stores the sorted intermediate table, sorts the second table, and finally merges the two to form the join result. Use the **`Suggest Sort Merge Join`** property on the SQL Properties pane to encourage a sort-merge. This property adds MAGIC=102 to the PROC SQL invocation, as follows: **`proc sql _method magic=102;`**.

Index Joins

An index join looks up each row of the smaller table by querying an index of the large table. When chosen by the optimizer, an index join usually outperforms a sort-merge join on the same data. To get the best join performance, you should ensure that both tables have indexes created on any columns that the you want to participate in the join relationship. The SAS SQL optimizer considers an index join when:

- The join is an equijoin in which tables are related by equivalence conditions on key columns.
- Joins with multiple conditions are connected by the AND operator.
- The larger table has an index composed of all the join keys.

Encourage an index nested loop with IDXWHERE=YES as a data set option, as follows: **`proc sql _method; select ...  from smalltable, largetable(idxwhere=yes)`**. You can also turn on the **`Suggest Index Join`** property on the properties panes for the tables in the query.

Hash Joins

The optimizer considers a hash join when an index join is eliminated as a possibility. With a hash join, the smaller table is reconfigured in memory as a hash table. SQL sequentially scans the larger table and performs row-by-row hash lookup against the small table to form the result set. A memory-sizing formula, which is not presented here, determines whether a hash join is chosen. The formula is based on the PROC SQL option BUFFERSIZE, whose default value is 64 KB. On a memory-rich system, consider increasing BUFFERSIZE to increase the likelihood that a hash join is chosen. You can also encourage a hash join by increasing the default 64 KB PROC SQL buffersize option. Set the **`Buffer Size`** property on the SQL Properties pane to **`1048576`**.

Setting the Implicit Property for a Join

Problem

You want to decide whether the **`Implicit`** property for a join should be enabled. This setting determines whether the join condition is processed implicitly in a WHERE statement or explicitly in a FROM clause in the SELECT statement.

Solution

You can access the **`Implicit`** property in the SQL Properties pane. You can also right-click a join in the **`Create`** tab to access the property in the pop-up menu. The following table depicts the settings available for each type of join, along with a sample of the join condition code generated for the join type:

Table 17.5 Implicit and Explicit Properties for SQL Join Types

Join Type	Join Condition Code

Inner	Can generate an implicit inner join condition in a WHERE statement near the end of the query:
	`where`
	`   POSTALCODES.Name = UNITEDSTATES.Name`
	You can use an implicit join only when the tables are joined with the equality operator. You can also generate an explicit inner join condition in a FROM clause in the SELECT statement:
	`from`
	`    srclib.POSTALCODES inner join`
	`            srclib.UNITEDSTATES`
	`               on`
	`               (`
	`                  POSTALCODES.Name = UNITEDSTATES.Name`
	`               )`

Full	Can generate an explicit join condition in a FROM clause in the SELECT statement:
	`from`
	`   srclib.POSTALCODES full join`
	`            srclib.UNITEDSTATES`
	`               on`
	`               (`
	`                  POSTALCODES.Name = UNITEDSTATES.Name`
	`               )`

Left	Can generate an explicit join condition in a FROM clause in the SELECT statement:
	`from`
	`   srclib.POSTALCODES left join`
	`            srclib.UNITEDSTATES`
	`               on`
	`               (`
	`                  POSTALCODES.Name = UNITEDSTATES.Name`
	`               )`

Join Type	Join Condition Code

Right	Can generate an explicit join condition in a FROM clause in the SELECT statement:
	`from`
	`   srclib.POSTALCODES right join`
	`            srclib.UNITEDSTATES`
	`               on`
	`               (`
	`                  POSTALCODES.Name = UNITEDSTATES.Name`
	`               )`

Cross	Can generate an explicit join condition in a FROM clause in the SELECT statement:
	`from`
	`   srclib.POSTALCODES cross join`
	`   srclib.UNITEDSTATES`

Union	Can generate an explicit join condition in a FROM clause in the SELECT statement:
	`from`
	`   srclib.POSTALCODES union join`
	`   srclib.UNITEDSTATES`

The **`Implicit`** property is disabled by default for all of the join types except the inner join.

Enabling Pass-Through Processing

Problem

You want to decide whether to enable pass-through processing, which sends DBMS-specific statements to a database management system and retrieves the DBMS data directly. In some situations, pass-through processing can improve the performance of the SQL Join transformation in the context of a SAS Data Integration Studio job. Pass-through processing is enabled with options that are found on the SQL Properties pane and the properties panes for the tables found in SAS queries. However, its impact can be sufficient enough to merit a dedicated topic.

Solution

You can use the **`Pass Through`** property on the SQL Join Properties pane to determine whether explicit pass-through processing is used. When the **`Pass Through`** property is set to **`Yes`**, you can send DBMS-specific statements to a database management system and retrieve DBMS data directly, which sometimes is faster than processing the SQL query on the SAS system. When **`Pass Through`** is set to **`No`**, explicit pass-through processing will not be used.

Tasks

Explicit Pass-Through Processing

Explicit pass-through is not always feasible. The query has to be able to work as is on the database. Therefore, if the query contains anything specific to SAS beyond the outermost select columns portion, the database will generate errors. For example, using any of the following in a WHERE clause expression or in a subquery on the WHERE or FROM clauses will cause the code to fail on the database if pass through is set to **`Yes`**:

- SAS formats
- SAS functions
- DATE or DATETIME literals or actual numeric values
- date arithmetic (usually will not work)
- INTO: macro variable
- data set options

The SQL Properties pane also contains the **`Target Table is Pass Through`** property, which determines whether explicit pass-through is active for the target table. This property enables the target to have the select rows inserted into the target within the explicit operation. This is valid only when all the tables in the query, including the target, are on the same database server. The **`Target Table is Pass Through`** property has a corresponding property, named **`Target Table Pass Through Action`**. The **`Truncate`** option in this property is useful for DBMS systems that will not allow the target to be deleted or created. In this case, the only option is removing all of the rows. If **`Truncate`** is selected, the table will have all of its rows deleted. If the table doesn't exist, it is created.

Implicit Pass-Through Processing

Even if **`Pass Through`** is set to **`No`**, PROC SQL will still try to pass the query or part of the query down to the database with implicit pass-through. This attempt to optimize performance is made without the user having to request it. SQL implicit pass-through is a silent optimization that is done in PROC SQL. Implicit pass-through interprets SAS SQL statements, and, whenever possible, rewrites the SAS SQL into database SQL.

There is no guarantee that the SQL will be passed to the database. However, PROC SQL will try to generate SQL that will pass to the database. If the optimization succeeds in passing a query (or parts of a query) directly to a database, the SQL query executes on the database and only the results of the query are returned to SAS. This can greatly improve the performance of the PROC SQL code. If the query cannot be passed to the database, records are read and passed back to SAS, one at a time. Implicit pass-through is disabled by the following query constructs:

- Heterogeneous queries: Implicit pass-through will not be attempted for queries that involve different engines or on queries that involve a single engine with multiple librefs that cannot share a single connection because they have different connection properties (such as a different **`database= value`**). You can use the **`Pass Through`** property to run these queries with explicit pass-through processing. You can also use the **`Upload Library Before SQL`**, **`Pre-Upload Action`**, and **`Use Bulkload for Upload`** properties in the table properties panes to improve the situation.

  Note: The **`Upload Library Before SQL`** property can be used to create a homogeneous join, which then can enable an explicit pass-through operation. This property allows you to select another library on the same database server as other tables in the SQL query. The best choice for a library would be a temporary space on that database server. The operations on that temporary table can also be modified to choose between deleting all rows or deleting the entire table. Bulk-load is also an option for the upload operation with the **`Use Bulkload for Uploading`** property. It is generally good practice to upload the smaller of the tables in the SQL query because this operation could be expensive. △

- Queries that incorporate explicit pass-through statements: If explicit pass-through statements are used, the statements are passed directly to the database as they are. Therefore, there is no need to try to prepare or translate the SQL with implicit pass-through to make it compatible to the database. It is already assumed compatible.
- Queries that use SAS data set options: SAS data set options cannot be honored in a pass-through context.
- Queries that use an INTO: clause: The memory associated with the host variable is not available to the DBMS processing the query. The INTO: clause is not supported in the SQL Join transformation.
- Queries that contain the SAS OUTER UNION operator: This is a non-ANSI SAS SQL extension.
- Specification of a SAS Language function that is not mapped to a DBMS equivalent by the engine. These vary by database.
- Specification of ANSIMISS or NOMISS in the join syntax.

Using Property Sheet Options to Optimize SQL Processing Performance

Problem

You want to set specific options in the SQL Properties pane or table properties panes located on the **`Designer`** tab in the properties window of an SQL Join transformation. These options are intended to improve the performance of SQL processes included in a SAS Data Integration Studio job.

Solution

Use one of the following techniques:

- Bulk load tables.
- Optimize the SELECT statement.
- Set buffering options.
- Use threaded reads.
- Write user-written code.

Tasks

Bulk Load Tables

The fastest way to insert data into a relational database when using the SAS/ACCESS engine is to use the bulk-loading capabilities of the database. By default, the SAS/ACCESS engines load data into tables by preparing an SQL INSERT statement, executing the INSERT statement for each row, and issuing a COMMIT. If you specify BULKLOAD=YES as a DATA step or LIBNAME option, the database load utility is invoked. This invocation enables you to bulk load rows of data as a single unit, which can significantly enhance performance. You can set the BULKLOAD option on the **`Bulkload to DBMS`** property pane for the target table. Some databases require that the table be empty in order to load records with their bulk-load utilities. Check your database documentation for this and other restrictions.

For smaller tables, the extra overhead of the bulk-load process might slow performance. For larger tables, the speed of the bulk-load process outweighs the overhead costs. Each SAS/ACCESS engine invokes a different load utility and uses different options. More information on how to use the bulk-load option for each SAS/ACCESS engine can be found in the online documentation for each engine.

The **`Use Bulkload for Uploading`** and **`Bulkload Options`** properties are available on the properties window for each table in a query. The **`Use Bulkload for Uploading`** property applies to the source table. It is a valid option only when the source table is being uploaded to the DBMS to create a homogeneous join. The **`Bulkload to DBMS`** property applies to target tables and turns bulk loading on and off. The **`Bulkload to DBMS`** property is not valid when the **`Target Table is Pass Through`** property on the SQL Properties pane is set to **`Yes`**.

The option to bulk load tables applies only to source tables that are participating in a heterogeneous join. Also, the user must be uploading the table to the DBMS where the join is performed.

Optimize the SELECT Statement

If you set the **`Select *`** property to **`Yes`** in the Select Properties pane, a Select * statement that selects all columns in the order in which they are stored in a table is run when the query is submitted. If you set the **`Select *`** property to **`No`** and enter only the columns that you need for the query in the SELECT statement, you can improve performance. You can also enhance performance by carefully ordering columns so that non-character columns (such as numeric, DATE, and DATETIME) come first and character columns come last.

Set Buffering Options

You can adjust I/O buffering. Set the **`Buffer Size`** property to 128 KB to promote fast I/O performance (or 64 KB to enhance large, sequential processes). The **`Buffer Size`** property is available in the SQL Properties pane. Other buffering options are database-specific and are available in the properties pane for each of the individual tables in the query. For example, you can set the READBUFF option by entering a number in the **`Number of Rows in DBMS Read`** property in the properties pane, which buffers the database records read before passing them to SAS. INSERTBUFF is an example of another option available on some database management systems.

You should experiment with different settings for these options to find optimal performance for your query. These are data set options; therefore, do not specify them unless you know that explicit pass-through or implicit pass-through will not be used on that portion of the query because they could actually slow performance. If these options are present in the query at all, they will prevent implicit pass-through processing. If these are options are present on the part that is being explicitly passed through, a database error will occur because the database won't recognize these options.

For example, if the **`Target Table is Pass Through`** property on the SQL Properties pane is set to **`Yes`**, then using INSERTBUFF data set options on this target table will cause an error on the database. If the **`Pass Through`** property in the SQL Properties pane is set to **`Yes`** and a number is specified in the **`Buffer Size`** property, the database will not recognize this option on the FROM clause of the query and return an error. One way around the risk of preventing implicit pass-through is specifying these options on the LIBNAME statement instead, but then it applies to all tables using that LIBNAME and to all access to those tables. That being said, these buffering data set options are great performance boosters if the database records will all be copied to SAS before the query runs in SAS (with no pass through) because it buffers the I/O between the database and SAS into memory. The default is 1, which is inefficient.

Use Threaded Reads

Threaded reads divide resource-intensive tasks into multiple independent units of work and execute those units simultaneously. SAS can create multiple threads, and a read connection is established between the DBMS and each SAS thread. The result set is partitioned across the connections, and rows are passed to SAS simultaneously (in parallel) across the connections. This improves performance.

To perform a threaded read, SAS first creates threads, which are standard operating system tasks controlled by SAS, within the SAS session. Next, SAS establishes a DBMS connection on each thread. SAS then causes the DBMS to partition the result set and reads one partition per thread. To cause the partitioning, SAS appends a WHERE clause to the SQL so that a single SQL statement becomes multiple SQL statements, one for each thread. The **`DBSLICE`** option specifies user-supplied WHERE clauses to partition a DBMS query for threaded reads. The **`DBSLICEPARM`** option controls the scope of DBMS threaded reads and the number of DBMS connections. You can enable threaded reads with the **`Parallel Processing with Threads`** property on the SQL Properties pane.

Write User-Written Code

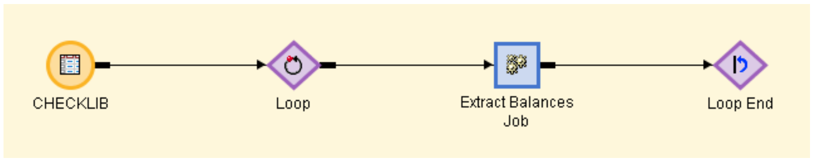
The **`User Written`** property determines whether the query is user-written or generated. When the **`User Written`** property on the SQL Properties pane is set to **`Yes`**, you can edit the code on the **`Source`** tab, and the entire job will be saved as user written. When the **`User Written`** property in the Where, Having, or Join Properties pane is set to **`Yes`**, you can then enter code directly into the field. Therefore, you can either write a new SQL query from scratch or modify a query that is generated when conditions are added to the top section of the **`Where/Having/Join`** tab. When **`User Written`** is set to **`No`** in any properties pane, the SQL field is read-only. It displays only the generated query. User-written code can be used as a last resort because the code can't be regenerated from the metadata when there are changes. The **`User Written`** property is available in the SQL Properties pane and in the Where/Having/Join Properties pane.

CHAPTER 18 Working with Iterative Jobs and Parallel Processing

About Iterative Jobs

An iterative job is a job with a control loop in which one or more processes are executed multiple times. For example, the following display shows the process flow for an iterative job.

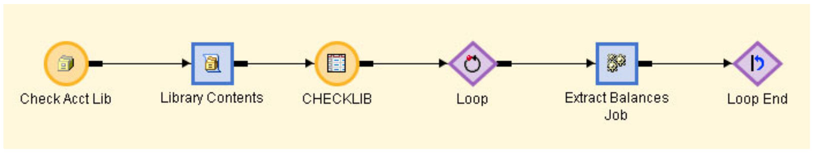
Display 18.1 Iterative Job



The process flow specifies that the inner Extract Balance job is executed multiple times, as specified by the Loop transformations and the CHECKLIB control table. The inner job is also called a *parameterized job* because it specifies its inputs and outputs as parameters.

The job shown in the previous example uses a control table that was created in a separate library contents job. This job created a control table that contains a static list of the tables included in the input library at the time that the job was run. You can also reuse an existing control table or create a new one. Many times, you will want to add the library input and the Library Contents transformation directly to an iterative job, as shown in the following example.

Display 18.2 Control Table Job in an Iterative Job



When the input library and the Library Contents transformation are added to the iterative job, the contents of the control table are dynamically generated each time that the iterative job is run. This arrangement ensures that the list of tables in the Check Acct Lib table is refreshed each time that the job is run. It also ensures that the tables are processed iteratively as each row in the control table is read.

Additional Information

For an example of how the steps in the iterative process are performed, see the "Example: Create an Iterative Job That Uses Parallel Processing" topic in SAS Data Integration Studio help.

Creating and Running an Iterative Job

Problem

You want to run a series of similarly structured tables through the same task or series of tasks. For example, you might need to extract specific items of census data from a series of 50 tables. Each table in the series contains data from one of the 50 states in the United States.

Solution

You need to create an iterative job that enables you to run a series of tables through the tasks contained in a job that is placed between Loop and Loop End transformations. This iterative job also contains a control table that lists the tables that are fed through the loop.

Tasks

Create and Run the Iterative Job

Perform the following steps to create and run the iterative job:

1 Create the control table and parameterized job that will be included in the iterative job.

2 Create an empty job.

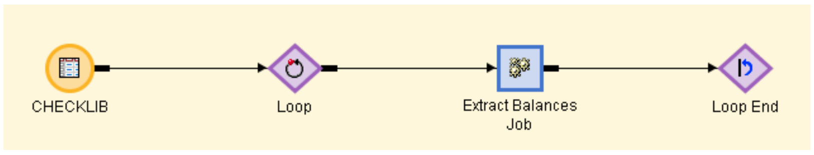
3 Drag the Loop End transformation from the Control folder in the Process Library tree. Then, drop it into the empty job.

4 Drag the parameterized job from the Project tree. Then, drop it into the input drop zone for the Loop End transformation.

5 Drag the Loop transformation from the Control folder in the Process Library tree. Then, drop it into the input drop zone for the parameterized job.

6 Drag the control table and drop it into the input drop zone for the Loop transformation. A sample completed iterative job is shown below.

Display 18.3 Completed Iterative Job



7 Open the **`Loop Options`** tab in the properties window for the Loop transformation. Select the **`Execute iterations in parallel`** check box. Also select the **`One process for each available CPU node`** check box in the **`Maximum number of concurrent processes`** group box.

8 Open the **`Parameter Mapping`** tab. Make sure that the appropriate value **`Source table`** field is mapped to the parameter listed in the **`Parameters`** field. The exact mapping depends on the columns that are included in the source table and the parameter that is set on the parameterized job.

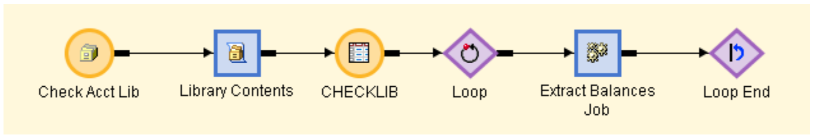
9 Close the properties window for the Loop transformation.

10 Run the iterative job.

Variation: Add the Input and Transformation Directly To a Job

You can customize the basic process by adding the library input and the Library Contents transformation directly to an iterative job, as shown in the following example.

Display 18.4 Control Table Job in an Iterative Job



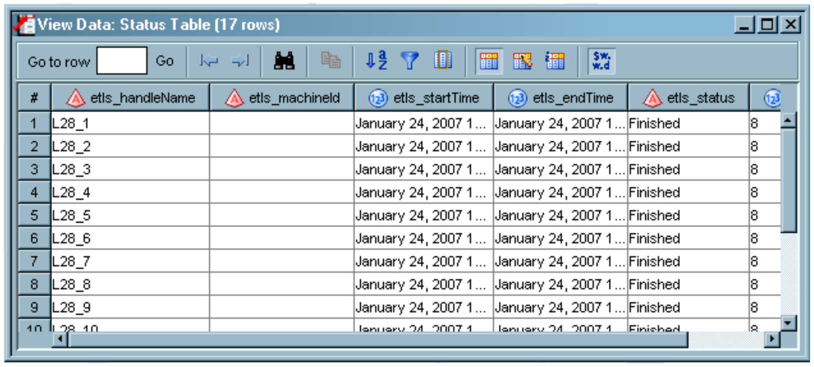
When the input library and the Library Contents transformation are added to the iterative job, the contents of the control table are dynamically generated each time that the iterative job is run. This arrangement ensures that the list of tables in the control table is refreshed each time that the job is run. It also ensures that the tables are processed iteratively as each row in the control table is read. For information about control table jobs, see "Creating a Control Table" on page 338.

Examine the Results

The output for the completed iterative processing is found in the output table for the parameterized job. In addition, the Loop transformation provides a status and run-time information in the temporary output table that is available when it is included in a submitted job. Once the job has completed without any errors, you can perform the following steps to review both the status data and iterative job output:

1 Right-click the Loop transformation and click **`View Data`**. A sample View Data window for the status information in the Loop transformation temporary output table is shown in the following example.

Display 18.5 Loop Transformation Temporary Output Table

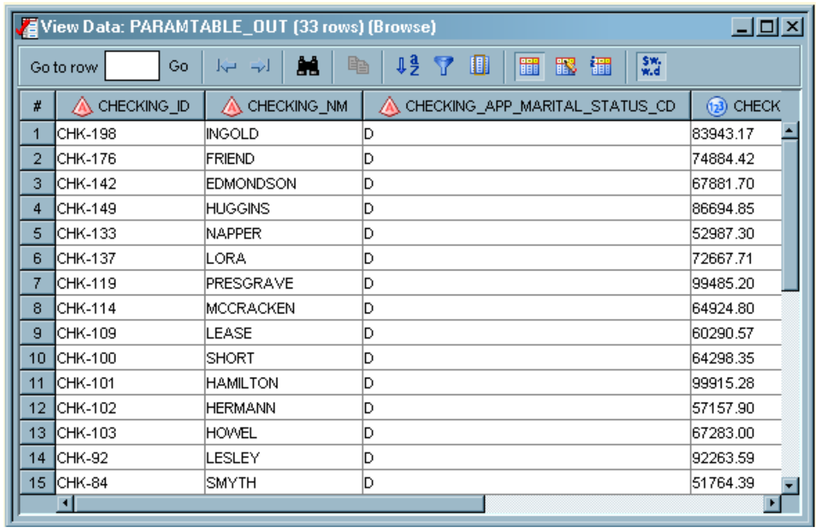


#	etls_handleName	etls_machineId	etls_startTime	etls_endTime	etls_status	
1	L28_1		January 24, 2007 1...	January 24, 2007 1...	Finished	8
2	L28_2		January 24, 2007 1...	January 24, 2007 1...	Finished	8
3	L28_3		January 24, 2007 1...	January 24, 2007 1...	Finished	8
4	L28_4		January 24, 2007 1...	January 24, 2007 1...	Finished	8
5	L28_5		January 24, 2007 1...	January 24, 2007 1...	Finished	8
6	L28_6		January 24, 2007 1...	January 24, 2007 1...	Finished	8
7	L28_7		January 24, 2007 1...	January 24, 2007 1...	Finished	8
8	L28_8		January 24, 2007 1...	January 24, 2007 1...	Finished	8
9	L28_9		January 24, 2007 1...	January 24, 2007 1...	Finished	8
10	L28_10		January 24, 2007 1	January 24, 2007 1	Finished	8

Each row in this table contains information about an iteration in the job.

2 Double-click the icon for the parameterized job. After the parameterized job opens, right-click the target table icon and click **`View Data`**. A sample View Data window for the iterative data is shown in the following example.

Display 18.6 View of Target Table Output



View Data: PARAMTABLE_OUT (33 rows) (Browse)

#	CHECKING_ID	CHECKING_NM	CHECKING_APP_MARITAL_STATUS_CD	CHECK
1	CHK-198	INGOLD	D	83943.17
2	CHK-176	FRIEND	D	74884.42
3	CHK-142	EDMONDSON	D	67881.70
4	CHK-149	HUGGINS	D	86694.85
5	CHK-133	NAPPER	D	52987.30
6	CHK-137	LORA	D	72667.71
7	CHK-119	PRESGRAVE	D	99485.20
8	CHK-114	MCCRACKEN	D	64924.80
9	CHK-109	LEASE	D	60290.57
10	CHK-100	SHORT	D	64298.35
11	CHK-101	HAMILTON	D	99915.28
12	CHK-102	HERMANN	D	57157.90
13	CHK-103	HOWEL	D	67283.00
14	CHK-92	LESLEY	D	92263.59
15	CHK-84	SMYTH	D	51764.39

Remember that you set a default value for the parameter on the output table when you set up the parameterized job. You can change the default value to see a different portion of the outputted data.

Additional Information

For detailed information about iterative jobs, see the “Example: Create an Iterative Job That Uses Parallel Processing” topic in the SAS Data Integration Studio help. For detailed information about specific usage issues, see the “Usage Notes for Iterative Jobs” topic in the SAS Data Integration Studio help.

Creating a Parameterized Job

Problem

You want to create a job that will enable you to perform an identical set of tasks on a series of tables. For example, you might need to extract specific demographic information for each of the 50 states in the United States when the data for each state is contained in a separate table.

Solution

You need to create a job that enables you to run each table through the loop in an iterative job. This job then writes data to an output table with each iteration. You set parameters on the job, the input table, and the output table. Then, you connect the parameters to the control table in the iterative job.

Tasks

Create the Parameterized Job

Perform the following steps to create the parameterized job:

1 Create and register the input and output tables.

2 Create an empty job.

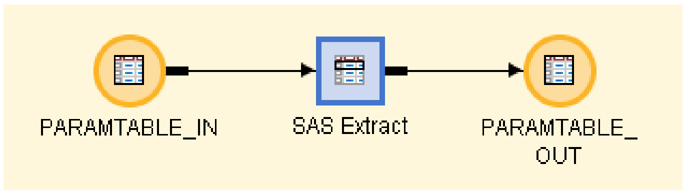
3 Drag the output table and drop it into the empty job. The output table must contain exactly the same columns as the tables listed in the control table for the loop processing in the iterative job to work properly.

4 Drag the transformation that will be used to process the job from the Process Library tree. Then, drop it into the input drop zone for the Loader transformation. Release the mouse button.

5 Drag the input table from the Project tree. Then, drop it into the input drop zone for the transformation just added to the Process Designer.

6 By default, a temporary output table and a Table Loader transformation are added to the Process Designer, but they are not needed for this job. Right-click the Table Loader transformation and select the **Delete** option. Click **OK** in the Confirm Remove window. The temporary output table and the Table Loader are deleted. A sample completed parameterized job is shown in the following example.

Display 18.7 Completed Parameterized Job



Set Input and Output Parameters

Perform the following steps to set the input and output table parameters for the parameterized job:

1 Open the **Parameters** tab in the properties window for the input table. Click **New** to display the Create Parameter window. Enter appropriate values in the following fields:

Parameter Name: a name for the macro variable, such as **Marital Status**.

Macro Variable Name: a valid macro variable name, such as **mstatus**.

Default value: a default value for the output table, such as **CHECKING_ACCOUNT_DIVORCED** in the **Default value** field. (You can enter the name of any tables listed in the control table list here.)

2 Close the Create Parameter window and open the **Physical Storage** tab. Enter an appropriate value in the **Name** field. Create this value by combining an ampersand sign with the value entered in the **Macro Variable Name** field on the Create Parameter window (for example, **&mstatus**). Close the properties window for the input table.

3 Open the **Parameters** tab in the properties window for the output table. Click **New** to display the Create Parameter window. Enter appropriate values in the following fields:

Parameter Name: a name for the macro variable, such as **Marital Status Out**.

Macro Variable Name: a valid macro variable name, such as **mstatus**.

Default value: a default value for the output table, such as **CHECKING_ACCOUNT_DIVORCED** in the **Default value** field. (You can enter the name of any tables listed in the control table list here.)

4 Close the Create Parameter window and open the **Physical Storage** tab. Enter an appropriate value in the **Name** field. Create this value by combining an ampersand sign with the value entered in the **Macro Variable Name** field on the Create Parameter window and appending **.OUT** to the combination (for example, **&mstatus.OUT**). Close the properties window for the output table.

Set Parameters for the Job

Perform the following steps to set the parameters for the parameterized job and to complete job configuration:

1 Open the **Parameters** tab in the properties window for the parameterized job.

2 Click **Import** to display the Import Parameters window. Click an appropriate value such as PARAMTABLE_IN in the **Available Parameters** field. Select the parameter assigned to the input table and move it to the **Selected Parameters** field. Then, close the properties window.

Complete Parameterized Job Configuration

Perform the following steps to complete the configuration of the parameterized job:

1 Configure any settings needed to process the data in the parameterized job. For example, you can set a WHERE condition in an Extract transformation if one is included in the job. These settings vary depending on the structure of the individual job.

2 Open the **Mapping** tab in the properties window for the transformation included in the parameterized job. Verify that all of the columns in the source table are mapped to an appropriate column in the target table and close the properties window.

3 Do not run the job. It will be submitted as a part of the iterative job.

Note: For detailed information about parameterized jobs, see the “Example: Create a Parameterized Job For Use In an Iterative Job” topic in the SAS Data Integration Studio help. △

Additional Information

For detailed information about iterative jobs, see the “Example: Create an Iterative Job That Uses Parallel Processing” topic in the SAS Data Integration Studio help. For detailed information about specific usage issues, see the “Usage Notes for Iterative Jobs” topic in the SAS Data Integration Studio help.

Creating a Control Table

Problem

You want to create a control table that lists the tables that you plan to include in an iterative job. Iterative jobs are used to run a series of similarly structured tables through the same task or series of tasks. The control table supplies the name of the table that is run through each iteration of the job.

Solution

You can reuse an existing control table or create one manually. You can also create a job that uses the Library Contents transformation. This transformation generates a listing of the tables contained in the library that holds the tables that you plan to run through the iterative job. This control table is based on the dictionary table of that library.

Tasks

Create and Register the Control Table

If you have an existing control table, you can use it. If you don't, you can use the Source Editor window in SAS Data Integration Studio to execute an SQL statement. The statement creates an empty instance of the table that has same column structure as the dictionary table for the library. Then use a Source Designer to register the empty table. Perform the following steps to create the empty control table:

1 Determine the identity and location of the library that contains the tables that you need to process in an iterative job.

2 From the SAS Data Integration Studio desktop, select **Tools ▶ Source Editor**.

   The Source Editor window appears. Submit code similar to the following code:

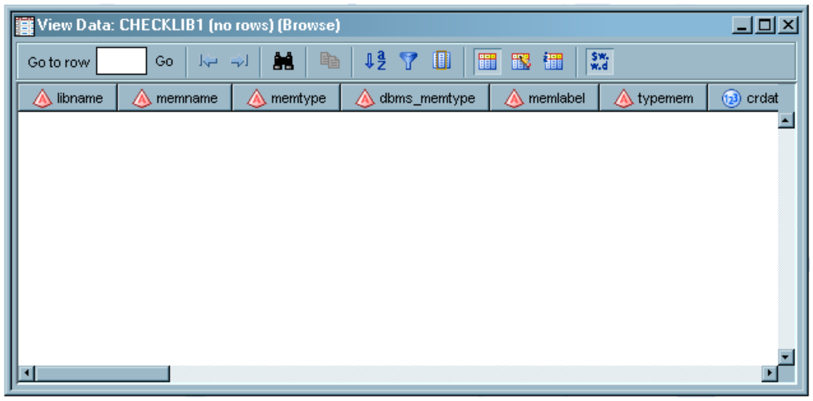
```
libname tgt 'C:\targets\sas1_tgt';

proc sql;
      create table tgt.CHECKLIB
      as select *
     from dictionary.tables
     where libname='checklib';
quit;
```

   Be sure to check the **Log** tab to verify that the code ran without errors.

3 Register the table that you just created using the Source Designer. This action creates a metadata object for the table.

4 (Optional) You can confirm that the empty control table was created in physical storage. Right-click the metadata object for the table and select **View Data**. A sample table is shown in the following example.

Display 18.8 View of Empty Control Table Output



Populate the Control Table Job

Perform the following steps to populate a control table job:

1 Create an empty job.

2 Drag the Library Contents transformation from the Access folder in the Process Library tree. Then, drop it into the empty job.

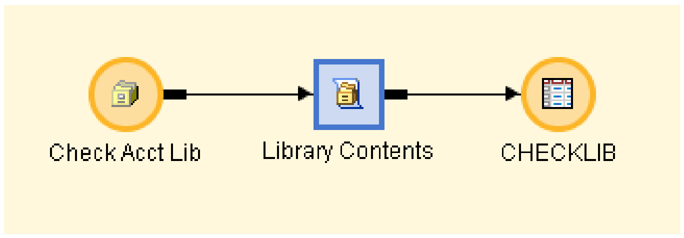
3 Drag the control table and drop it into the target table location in the Process Editor.

4 By default, a temporary output table and a Table Loader transformation are added to the Process Designer, but they are not needed for this job. Right-click the Table Loader transformation and select the **`Delete`** option. Click **`OK`** in the Confirm Remove window. The temporary output table and the Table Loader are deleted.

5 Open the **`Mapping`** tab in the properties window for the Library Contents transformation. Verify that all of the rows in the source table are mapped to the corresponding row in the target table and click **`Quick Map`** to correct any errors.

6 Drag the icon for the library that contains the tables that will be iteratively processed from Libraries folder in the Inventory tree. Then, drop the icon into the input zone for the Library Contents transformation. A sample completed control table job is shown in the example below.

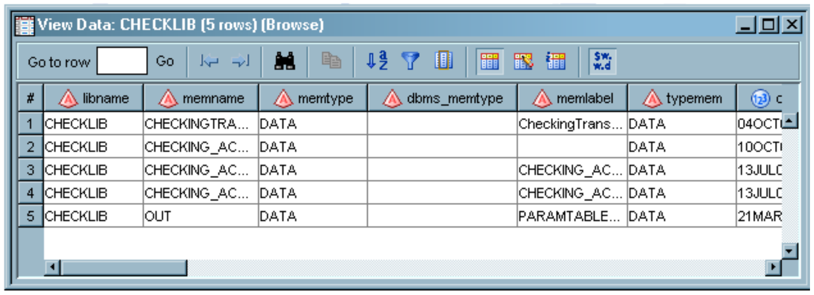
Display 18.9 Completed Control Table Job



7 Run the job.

8 If the job completes without error, right-click the control table icon and click **`View Data`**. The View Data window appears, as shown in the following example.

Display 18.10 View of Control Table Output



View Data: CHECKLIB (5 rows) (Browse)

#	libname	memname	memtype	dbms_memtype	memlabel	typemem	c
1	CHECKLIB	CHECKINGTRA...	DATA		CheckingTrans...	DATA	04OCT
2	CHECKLIB	CHECKING_AC...	DATA			DATA	10OCT
3	CHECKLIB	CHECKING_AC...	DATA		CHECKING_AC...	DATA	13JUL
4	CHECKLIB	CHECKING_AC...	DATA		CHECKING_AC...	DATA	13JUL
5	CHECKLIB	OUT	DATA		PARAMTABLE...	DATA	21MAR

Note that the all of the rows in the table are populated with the name of the control table in the libname column. This confirms that all of the rows are drawn from the appropriate library. You can now use the table as the control table for the iterative job.

For detailed information about control table jobs, see the "Example: Create and Populate the Control Table With a Library Contents Transformation" topic in the SAS Data Integration Studio help.

Additional Information

For detailed information about iterative jobs, see the "Example: Create an Iterative Job That Uses Parallel Processing" topic in the SAS Data Integration Studio help. For detailed information about specific usage issues, see the "Usage Notes for Iterative Jobs" topic in the SAS Data Integration Studio help.

About Parallel Processing

SAS Data Integration Studio uses a set of macros to enable parallel processing. You can enable these macros by doing one of the following:

- Selecting **`YES`** in the **`Enable parallel processing macros`** option on the **`Options`** tab of the properties window for a job.
- Including a Loop transformation in a job.

When you enable the parallel-processing option for a job, macros are generated at the top of the job code with comments to enable you to create your own transformations or code to take advantage of parallel processing.

When you include a Loop transformation in a job, the transformation generates the necessary macros to take advantage of sequential execution, symmetric multiprocessing (SMP) execution, or execution on a grid computing network.

No special software or metadata is required to enable parallel processing on SMP servers. If grid options have been enabled for a job, but the grid software has not been configured and licensed, SAS Data Integration Studio does not generate grid-enabled code for the job. It generates code that is appropriate for SMP on the SAS Application Server.

The following table describes the prerequisites that are required to enable parallel processing for SAS Data Integration Studio jobs. For details about these prerequisites, see the appropriate section in the documentation mentioned below.

Table 18.1 Prerequisites for Parallel Processing of SAS Data Integration Studio Jobs

Computers Used for Parallel Processing	Requirements
SMP machine with one or more processors	Specify a SAS®9 Workspace server in the metadata for the default SAS application server for SAS Data Integration Studio. See the "Specifying Metadata for the Default SAS Application Server" topic in SAS Data Integration Studio help.
Grid computing network	Specify an appropriate SAS Metadata Server to get the latest metadata object for a grid server. See the SAS Data Integration Studio chapter in the *SAS Intelligence Platform: Desktop Application Administration Guide*. Specify an appropriate SAS®9 Workspace Server in the metadata for the default SAS application server. Grid software must be licensed. Define or add a Grid Server component to the metadata that points to the grid server installation. The controlling server machine must have both a Grid Server definition and a SAS Workspace Server definition as a minimum to be able to run your machines in a grid. It is recommended that you also have the SAS Metadata Server component accessible to the server definition where your grid machines are located. Install Platform Computing software to handle workload management for the grid. *Note:* For additional information on these requirements, see the grid chapter in the *SAS Intelligence Platform: Application Server Administration Guide*. △

Setting Options for Parallel Processing

Problem

You want to take advantage of parallel processing and grid processing in SAS Data Integration Studio jobs.

Solution

If you need to enable parallel or grid processing for all jobs, set global options on the **Options** tab of the Options window for SAS Data Integration Studio. If you need to enable parallel or grid processing for a single iterative job, set the options available on the **Loop Options** tab of the properties window for the Loop transformation.

Tasks

The following tables describe how to set options for parallel processing and grid processing in SAS Data Integration Studio jobs.

Table 18.2 Global Options (affects all new jobs)

Option	Purpose	To Specify
Enable parallel processing macros for new jobs	Adds parallel processing macros to the code that is generated for all new jobs.	Select Tools ▶ Options from the menu bar. Click the **Options** tab. Specify the desired option.
Various grid computing options	Sets grid computing options for all new jobs.	Select Tools ▶ Options from the menu bar. Click the **Options** tab. Specify the desired option.

Table 18.3 Local Options (affects the current job or transformation)

Option	Purpose	To Specify
Enable parallel processing macros	When **YES** is selected, this option adds parallel processing macros to the code that is generated for the current job. Parallel processing macros are always included in the code that is generated for a Loop transformation.	Open the **Options** tab in the properties window for the job. Select **YES** or **NO** in the field for this option.
Various grid computing options for the Loop transformation	Sets grid options for the current Loop transformation	Open the **Loop Options** tab in the properties window for the Loop transformation. Specify the desired option.

Additional Information

For details about the global options included on the **Code Generation** tab of the Options window, see the description of the options in the “Code Generation Tab” topic in SAS Data Integration Studio help. For information about the options available on the **Loop Options** tab in the Loop transformation properties window, see the “About Loop Transformations” and “Loop Options Tab” topics. For information about specific usage issues, see the “Usage Notes for Parallel Processing” topic.

CHAPTER 19

Working with Slowly Changing Dimensions

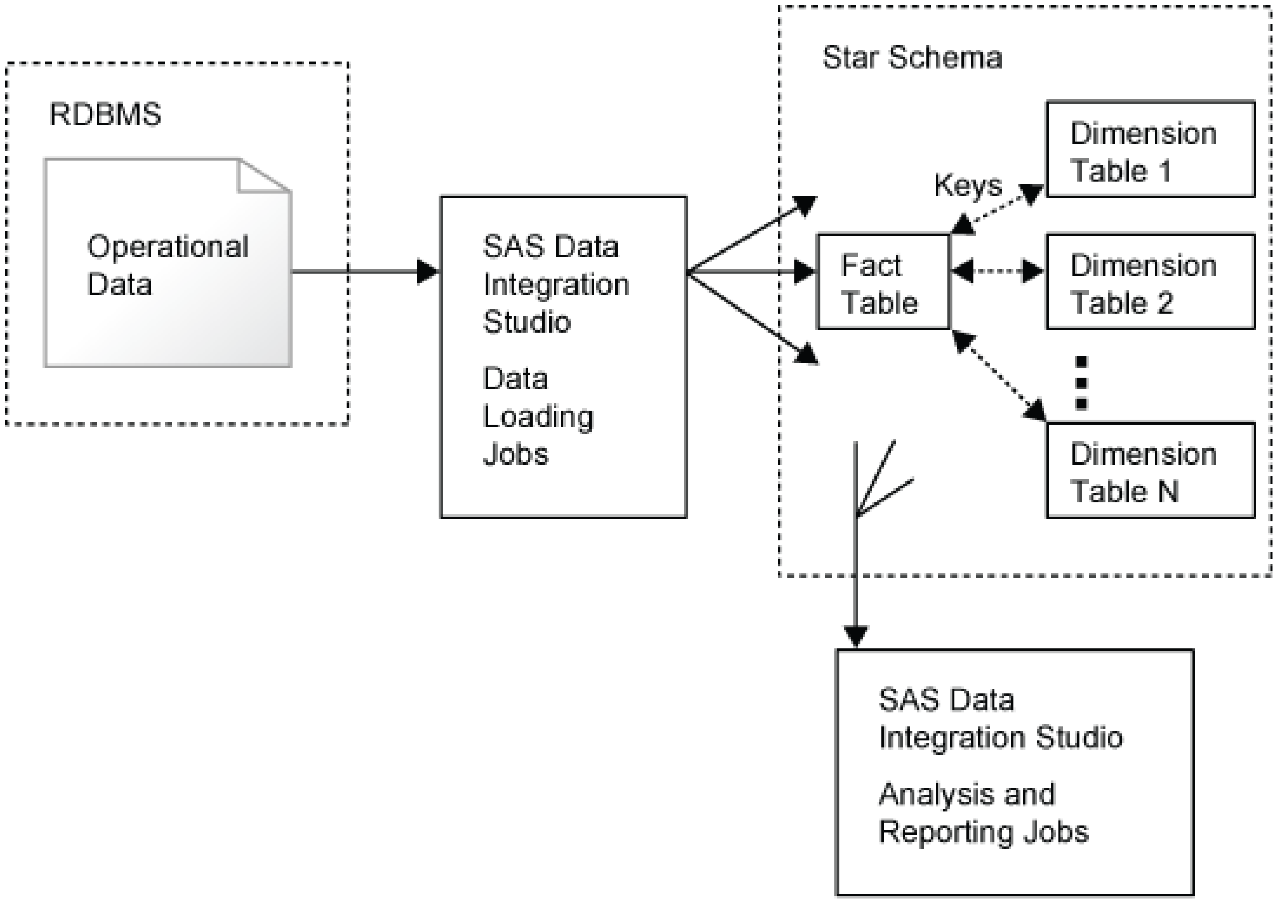
About Slowly Changing Dimensions (SCD)

SCD and the Star Schema

The star schema is an architecture that separates operational data into two categories: factual events and the detail data that describes those events. Numerical data about events is stored in a fact table. Character data that describes the events is stored in dimension tables. In the fact table, numerical values called keys identify the detail data in the dimension tables that is associated with each event.

As shown in the following diagram, SAS Data Integration Studio enables you to create jobs that load data into star schemas. Other jobs extract knowledge from the star schema.

Figure 19.1 The Star Schema and SAS Data Integration Studio



Dimension tables provide collection points for categories of information. Typical categories are customers, suppliers, products, and organizations.

To provide analytical power, type 2 slowly changing dimensions is implemented in dimension tables to retain a record of changes to data over time. Analysis of the record of data changes provides knowledge. For example, analysis of a dimension table that contains customer information might allow buying incentives to be offered to customers that are most likely to take advantage of those incentives.

In SAS Data Integration Studio, the process of loading dimension tables and maintaining the record of data changes takes place in the transformation called the SCD Type 2 Loader. Other transformations handle fact table loading and other aspects of star schema management, as described in "Transformations That Support Slowly Changing Dimensions" on page 349.

Three types of slowly changing dimensions are commonly defined, as described in the following subsections.

Type 1 SCD

Type 1 SCD stores one row of data for each member of a dimension. (Members are individuals with a unique ID number.) New data overwrites existing data, and a history of data changes is not maintained. For example, in a dimension table that contains customer information, if a row contains data for Mr. Jones, and if a new home address for Mr. Jones is included in an update of the dimension table, then the new address overwrites the old address.

Type 2 SCD

You can identify dimension tables as type 2 SCD when you see multiple rows of data per member. One row is the current row, which is the latest set of information for each member in the dimension. The other rows are said to be closed out, which means that they are no longer the current row. The closed–out rows maintain a historical record of changes to data.

Type 3 SCD

Type 3 SCD retains a limited history of data changes using separate columns for different versions of the same value As in Type 1 SCD, each member of the dimension is represented by a single row in the dimension table. An example of a type 3 SCD table might contain three columns for postal codes. The column names might be Current Postal Code, Previous Postal Code, and Oldest Postal Code. As dimension table updates add new postal codes, the values in the Current column move into the Previous column, and values in Previous column move into the Oldest column.

About the Star Schema Loading Process

The process for loading a star schema for slowly changing dimensions follows these general steps:

1 Stage operational data. In this initial step you capture data and validate the quality of that data. Your staging jobs make use of the Data Validation transformation, along with other data quality transformations and processes.

2 Load dimension tables. Data from the staging area is moved into the dimension tables of the star schema. Dimension tables are loaded before the fact table in order to generate the primary key values that are needed in the fact table.

3 Load the fact table. In this final step you run a job that includes the Lookup transformation, which loads numerical columns from the staging area into the fact table. Then the Lookup transformation captures foreign key values from the dimension tables.

About Type 2 SCD Dimension Tables

Dimension tables that are loaded with the SCD Type 2 Loader consist of a primary key column, a business key column, one or two change tracking columns, and any number of detail data columns. The primary key column is often loaded with values that are generated by the transformation. The business keys are supplied in the source data. Both the business key and the primary key can be defined to consist of more than one column, as determined by the structure of the source data.

Change tracking columns can consist of begin and end datetime columns, a version number column, or a current-row indicator column.

Begin and end datetime values specify the period of time in which each row was the current row for that member. The following diagram shows how data is added to begin and end datetime columns. Note how the end value for one row relates to the begin value for the row that superseded it. The end value for the current row is a placeholder future date.

Figure 19.2 Structure of an SCD Dimension Table

	Business Key (member ID)	Begin Datetime	End Datetime	Primary Key (generated)	Detail Data A	Detail Data B
current row	2138	27JUL2007	01JAN2599	25		
closed-out row	2138	15MAY2007	27JUL2007	18		
closed-out row	2138	22FEB2007	15MAY2007	6		

Tracking changes by version number increments a counter when a new row is added. The current row has the highest version number for that member. The version number for new members is current_version_number + 1.

Tracking changes using a current–row indicator column loads a 1 for the current row and 0s for all of the other rows that apply to that same member.

The preceding diagram shows a primary key column, the values for which are generated by the SCD Type 2 Loader. The generated primary key is necessary in order to uniquely identify individual rows in the dimension table. The generated primary key values are loaded into the star schema's fact table as foreign keys, to connect factual or numerical events to the detail data that describes those events.

About Change Detection and the Loading Process for SCD Dimension Tables

In jobs that run the SCD Type 2 Loader transformation, the dimension table loading process repeats the following process for each source row:

1 Compare the business key of the source row to the business keys of all of the current rows in the dimension table. If no match is found, then the source row represents a new member. The source row is written to the target and the loading process moves on to the next source row.

2 If the business key in the source matches a business key in the target, then specified detail data columns are compared between the matching rows. If no

differences in data are detected, then the source row is a duplicate of the target row. The source row is not loaded into the target and the loading process moves on to the next source row.

3 If business keys match and data differences are detected, then the source row represents a new current row for that member. The source row is written to the target, and the previous current row for that member is closed-out. To close out a row, the change tracking column or columns are updated as necessary, depending on the selected method of change tracking.

To learn how to use the SCD Type 2 Loader, see "Loading a Dimension Table Using Begin and End Datetime Values" on page 349.

About Cross-Reference Tables

During the process of loading an SCD dimension table, the comparison of incoming source rows to the current rows in the target is facilitated by a cross-reference table. The cross-reference table consists of all of the current rows in the dimension table, one row for each member. The columns consist of the generated key, the business key and a digest column named DIGEST_VALUE.

The digest column is used to detect changes in data between the source row and the target row that has a matching business key. DIGEST_VALUE is a character column with a length of 32. The values in this column are encrypted concatenations of the data columns that were selected for change detection. The encryption uses the MD5 algorithm, which is described in detail at **`http://www.faqs.org/rfcs/rfc1321.html`**.

If a cross-reference table exists and has been identified, it will be used and updated. If a cross-reference table has not been identified, then a new temporary table is created each time you run the job.

Cross-reference tables are identified in the **`Options`** tabs of the following transformations: SCD Type 2 Loader and Key Effective Date, in the field **`Cross-Reference Table Name`**.

About the Structure and Loading of Fact Tables

Fact tables contain numerical columns that describe events, along with foreign-key columns that identify detail data that describes the events. The detail data is stored in dimension tables.

Fact tables are loaded with the Lookup transformation. Input data for the transformation consists of a source table that contains the numeric columns and the dimension tables that contain the foreign-key columns. Because the dimension tables need to have primary keys assigned, you need to load your dimension tables before you load your fact table. The primary keys in the dimension table become the foreign keys in the fact table.

For each dimension table, you configure the Lookup transformation to find a match between specified columns in the source data and the dimension table. If a match is found, then the primary key of that dimension table row is loaded into the fact table as a foreign key.

The Lookup transformation also enables you to flexibly configure responses to exceptions and errors, so that you can maintain the quality of the data in your star schema.

To learn how to use the Lookup transformation, see "Loading a Fact Table" on page 353.

About Keys

In a star schema, key values connect fact tables to dimension tables. The following types of keys are used in the SCD Type 2 Loader:

primary keys
: uniquely identify the rows in fact and dimension tables. Primary key columns are identified in the **Keys** tab of the dimension table's Properties dialog box.

foreign keys
: connect rows in fact tables to rows in dimension tables. Foreign key values in a fact table are primary key values in the dimension tables. Foreign keys are loaded into fact tables using the Lookup transformation.

business keys
: uniquely identify the members in dimension tables. These keys are essential to Type 2 SCD, because each member can be represented by more than one row. They're called business keys because the values are supplied in the source table, as delivered from the operational system that collected the data. An example of a business key column might be named Customer_ID. This column can be loaded into the Customer dimension table to identify all of the rows (current and closed–out) that are associated with each customer. Business keys are identified in **Business Key** tab of the SCD Type 2 Loader's Properties dialog box.

generated keys
: establish key values for in dimension tables. The SCD Type 2 Loader can generate surrogate, retained, and unique keys.

surrogate keys
: are integer values that are generated in the SCD Type 2 Loader to provide primary or unique key values in dimension tables. By default, the generated values increment the highest existing number in a specified key column. You have the option of using an expression to modify the default generated value. You can also specify a starting point for each load using a lookup column.

retained keys
: consist of a numeric column of generated values that is combined with a datetime column to make up the primary key of a dimension table. The keys are said to be retained because the generated numeric values do not change after they have been assigned to a member. When a member is updated, the retained value is copied into the new row from the previous current row. The datetime value uniquely identifies the row.

Keys of any type can be defined to include multiple columns in your dimensional target table, as needed according to the structure of your source data.

You identify key columns in the **Keys** tab of the target table's Properties dialog box.

About Generated Keys

The SCD Type 2 Loader enables you to generate key values when you load a dimension table. The generated values are used as primary keys. After the keys are generated and the dimension table has been loaded, the primary key column is added to the fact table as a foreign key for that dimension.

In the **Generated Keys** tab of the SCD Type 2 Loader, you can configure a simple surrogate key that increments the highest existing value in a specified column. You can also use an expression to generate values in other than simple increments. To specify a unique starting point for the keys that are generated in each load, you can specify a lookup column.

In addition to surrogate keys, you can also generate retained keys. Retained keys provide a primary key value that consists of two columns, one containing a datetime value and the other containing a generated value. The generated value is retained because a single generated value is used for all of the rows that apply to a given member. The datetime value differentiates the rows.

As with surrogate keys, you can generate retained key values using expressions and lookup columns.

To enhance performance, you should create an index for your generated key column. If you identify your generated key column as the primary key of the table, then the index is created automatically. Surrogate keys should receive a unique or simple index consisting of one column. Retained keys should receive a complex index that includes the generated key column and the datetime column.

To create an index, open the Properties dialog box for the table and use the **`Index`** and **`Keys`** tabs.

Transformations That Support Slowly Changing Dimensions

SAS Data Integration Studio provides the following transformations that you can use to implement slowly changing dimensions:

SCD Type 2 Loader
: loads dimension tables, detects changes, tracks changes, and generates key values.

Lookup
: loads source data into fact tables and loads foreign keys from dimension tables, with configurable error handling. The lookup process accesses dimension tables using hash objects for optimal performance.

Fact Table Lookup
: loads source data into fact tables using key values from dimension tables. The lookup process uses SAS formats rather than the more efficient hash objects used in the Lookup transformation.

Key Effective Date
: updates dimension tables based on changes to the business key, when change detection is unnecessary.

Surrogate Key Generator
: generates unique key numbers for dimension tables, in a manner that is similar but less feature-rich than the SCD Type 2 Loader transformation. Use the Surrogate Key Generator when key generation is the sole task that is required at that point in the job.

Loading a Dimension Table Using Begin and End Datetime Values

Problem

You want to load data into a dimension table and maintain a history of data changes using begin and end datetime values.

Solution

Use the SCD Type 2 Loader to load datetime values from the source, or generate datetime values in specified formats.

Tasks

Perform the following steps to load an SCD dimension table that tracks data changes using begin and end datetime values:

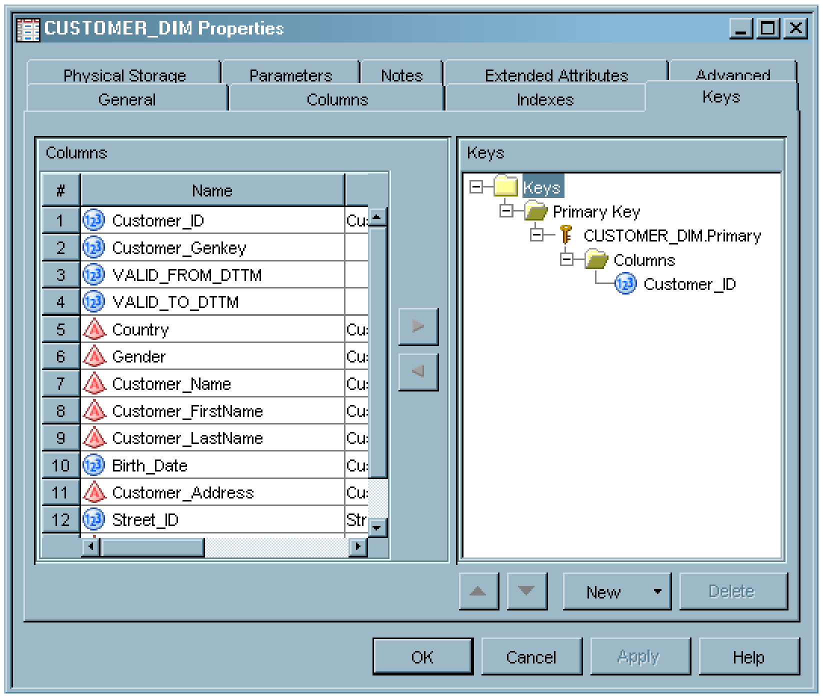
1 Create a job that includes the SCD Type 2 Loader transformation, a source table with a business key, and a target table that has numeric columns for begin and end datetime values.

2 In the Process Editor, double-click the target table to display the Properties dialog box. If you intend to generate primary key values, display the **`Columns`** tab and add a numeric column.

3 Click **`Apply`**, and then open the **`Keys`** tab. Right-click **`New`** and select **`Primary Key`**.

4 In the **`Columns`** list of the **`Keys`** tab, select the column or columns that make up your primary key and click the right arrow. The primary key column or columns appear in the **`Keys`** pane, as shown in the following display.

Display 19.1 Target's Primary Key Specification



Columns

#	Name	
1	Customer_ID	Cu
2	Customer_Genkey	
3	VALID_FROM_DTTM	
4	VALID_TO_DTTM	
5	Country	Cu
6	Gender	Cu
7	Customer_Name	Cu
8	Customer_FirstName	Cu
9	Customer_LastName	Cu
10	Birth_Date	Cu
11	Customer_Address	Cu
12	Street_ID	Str

Keys

```
Keys
  Primary Key
    CUSTOMER_DIM.Primary
      Columns
        Customer_ID
```

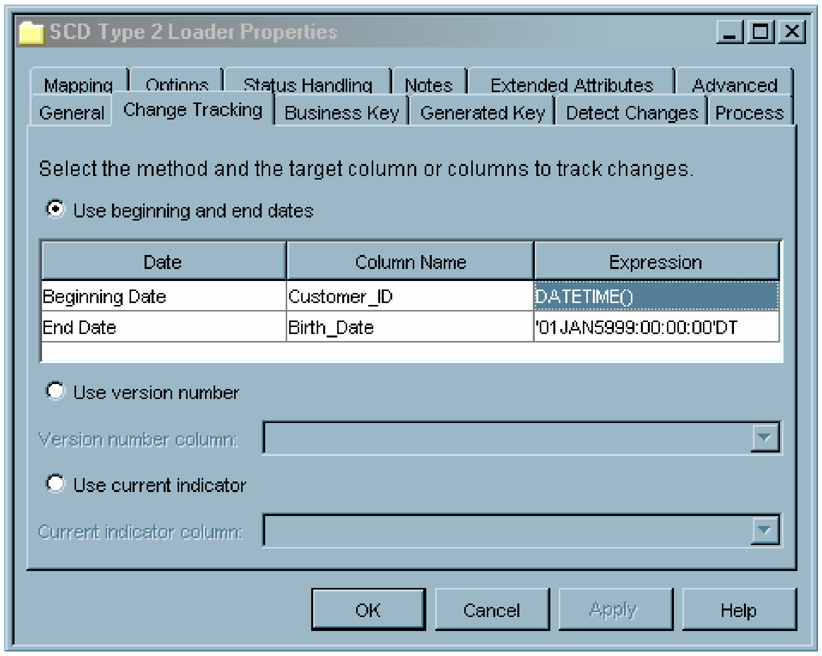
5 Click **`OK`** to save changes and close the Properties dialog box.

6 Open the Properties dialog box for the **`SCD Type 2 Loader`**.

7 Open the **`Mapping`** tab. For the business key and detail data columns in the source, click in the source column and drag to a target column to create arrows that connect the source and target columns. If you plan to load datetime values from the source, also add connecting arrows for those columns.

8 Click **`Apply`**, and then open the **`Change Tracking`** tab. Triple-click under **`Column Name`** to select numeric datetime columns from drop-down lists. By default, the datetime columns are the first two numeric columns in your target table.

Display 19.2 Typical Default Values in the Change Tracking Tab



Select the method and the target column or columns to track changes.

(●) Use beginning and end dates

Date	Column Name	Expression
Beginning Date	Customer_ID	DATETIME()
End Date	Birth_Date	'01JAN5999:00:00:00'DT

( ) Use version number

Version number column:

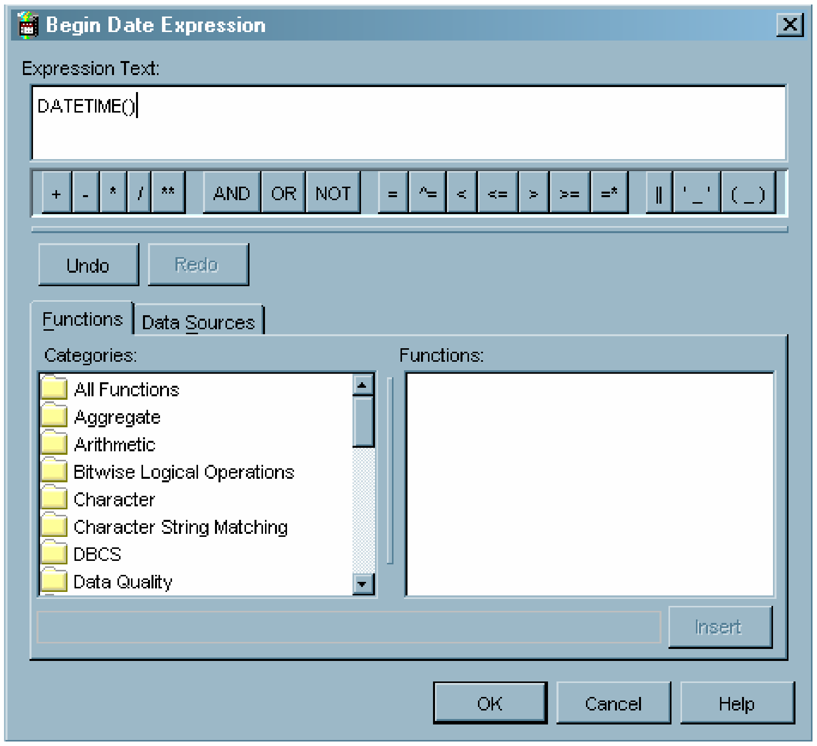
( ) Use current indicator

Current indicator column:

9 If you plan to load datetime values from the source, realize that the values in the two **`Expression`** fields will be used when datetime values are missing in the source. You might wish to change the default expressions to better suit the datetime format in your source data.

To modify the expressions with the Expression Builder, double click under **`Expression`** and click the button that appears in the field. The Expression Builder enables you to use logical operators, SAS functions, and a variety of datetime formats.

Display 19.3 The Expression Builder



10 When you're ready, click **`Apply`** and open the **`Options`** tab. Click **`Help`** to review the valid values for **`Format Type for Date`**. Specify the value that suits the nature of your source data. Then click **`Apply`** to save your changes.

11 If you plan to generate datetime values for change tracking, examine the default expressions in the **`Change Tracking`** tab.

12 Click **`Apply`**, and then open the **`Business Key`** tab.

13 In the **`Business Key`** tab, change the default values as necessary to specify the numeric source column or columns that make up the business key. If your business key consists of multiple columns, be sure to specify at the top of the list the column that provides unique member identifiers. If your business key uses a combination of columns to establish uniqueness, consider that the columns will be evaluated by the transformation in top-down order as listed in the **`Business Key`** tab.

14 Click **`Apply`**, and then open the **`Change Detection`** tab. In that tab you will see a list of all target columns other than those that those that are used for change tracking, the business key, or any generated key. To specify the columns that you include in change tracking, select the columns and click the right arrow. To optimize performance, select only the columns that you intend to analyze, and select shorter columns over longer columns when possible.

15 Click **`OK`** to save changes and close the Properties dialog box.

16 Review column mappings in the SCD Type 2 Loader and the target.

17 With the cursor in the Process Editor, right-click and select **`Save`** to save the changes to your job.

18 Click the **`Submit Source`** button in the toolbar to run the job.

Loading a Dimension Table Using Version Numbers or Current-Row Indicators

Problem

Your Type 2 SCD dimension table needs to use a means of tracking data changes other than begin and end datetime columns.

Solution

Use the **`Change Tracking`** tab in the SCD Type 2 Loader to generate version numbers or current-row indicators to identify the current row for each member in the dimension.

Tasks

Tracking changes with version numbers generates and loads a sequential number in a specified column. The advantages of version numbering are that the previous current row doesn't need to be updated, and a fully ordered list of entries is maintained. The current row for a given member is the row that has the highest version number.

The current-row indicator uses a 1 in the target column for all current rows and a 0 in the same column for all closed-out rows. This method of change tracking is useful if you have other means of ordering closed-out rows.

To implement version numbers or current-row indicators, follow the steps in "Loading a Dimension Table Using Begin and End Datetime Values" on page 349. In the target table, define one target column for change tracking instead of two. In the **`Change Tracking`** tab of the SCD Type 2 Loader, select version numbering or current-row indicators instead of begin and end datetime values.

Loading a Fact Table

Problem

You want to load data into a fact table and add foreign keys to connect the fact table to the dimension tables in the star schema.

Solution

Create a job that uses the Lookup transformation, and run that job after you run the jobs that load your dimension tables. The Lookup transformation loads fact columns from a source table and loads foreign key values from dimension tables. The generated primary key in the dimension table is loaded into the fact table when the business key in the fact table matches the business key in the dimension table.

Tasks

Perform the following steps to load a fact table:

1 Before you build the job that loads your fact table, be sure to first submit the jobs that load your dimension tables, as described in “Loading a Dimension Table Using Begin and End Datetime Values” on page 349.

2 Create a job in the Process Editor that includes a Lookup transformation.

3 Drop the source table into the source table drop box. The source table contains the fact columns and business key columns.

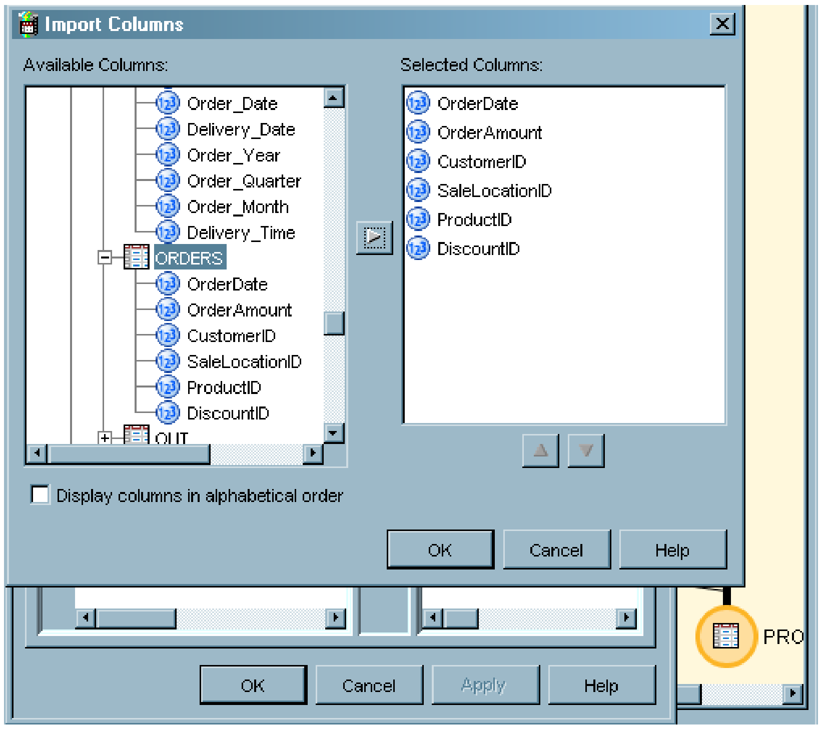
4 Drop one of the dimension tables onto the lookup table drop box. The dimension tables contain the key values that you want to load into your fact table.

5 To add drop boxes for additional dimension tables, right-click the Lookup transformation and click **`Add Input`**.

6 Drop dimension tables onto the lookup table drop boxes.

7 In the Lookup transformation, open the **`Mapping`** tab, right-click, and select **`Import`**. Import the fact data columns from the source table, as shown in the following display.

Display 19.4 Importing Source Columns into the Lookup Transformation


Import Columns

Available Columns:
- Order_Date
- Delivery_Date
- Order_Year
- Order_Quarter
- Order_Month
- Delivery_Time
- ORDERS
  - OrderDate
  - OrderAmount
  - CustomerID
  - SaleLocationID
  - ProductID
  - DiscountID
- OUT

Selected Columns:
- OrderDate
- OrderAmount
- CustomerID
- SaleLocationID
- ProductID
- DiscountID

☐ Display columns in alphabetical order

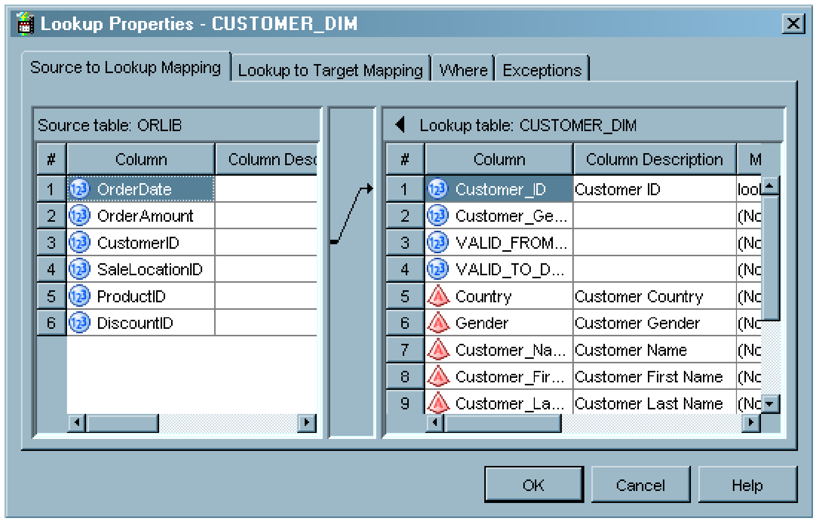
8 Right-click in **`Selected Columns`** and select **`Quick Map`** to draw connecting arrows between the columns in the source and the columns in the work table of the Lookup transformation.

9 To add foreign key columns, repeat the column import process for each of the lookup (dimension) tables. There’s no need to add mapping lines for these columns. Click **`Apply`**, and then open the **`Lookup`** tab.

10 In the **Lookup** tab, note that the first lookup table is highlighted. For this table you will define the matching condition between the dimension table and the source table. The matching condition specifies when a foreign key value is to be loaded into the target from the selected dimension table. The compared values are the business key columns. If the business key values match, the foreign key value is loaded into the target from the dimension table.

To set up the match between the source and the dimension table, click **Lookup Properties**. In the **Source to Lookup Mapping** tab, left-click on the business key column in the source and drag across to the business key column in the lookup table. Release the mouse to display a connecting arrow between the two columns, as shown in the following display:

Display 19.5 Defining the Match Between Business Key Columns



Lookup Properties - CUSTOMER_DIM

Source to Lookup Mapping | Lookup to Target Mapping | Where | Exceptions

Source table: ORLIB

#	Column	Column Desc
1	OrderDate	
2	OrderAmount	
3	CustomerID	
4	SaleLocationID	
5	ProductID	
6	DiscountID	

Lookup table: CUSTOMER_DIM

#	Column	Column Description	M
1	Customer_ID	Customer ID	lool
2	Customer_Ge...		(Nc
3	VALID_FROM...		(Nc
4	VALID_TO_D...		(Nc
5	Country	Customer Country	(Nc
6	Gender	Customer Gender	(Nc
7	Customer_Na...	Customer Name	(Nc
8	Customer_Fir...	Customer First Name	(Nc
9	Customer_La...	Customer Last Name	(Nc

11 If you want to define a WHERE clause that further refines the match between the business key columns, click the **Where** tab and build an expression. Click **Apply** to save changes.

Note: If you use a WHERE clause, and if the lookup table uses a generated key, you can improve performance by creating an index on the generated key column, as described in "About Generated Keys" on page 348. △

12 To define the lookup table columns that will be loaded into the fact table when a match occurs between business key values, open the **Lookup to Target Mapping** tab, left-click in the lookup column, and drag across to the target column. Release the mouse to display a connecting arrow between the columns.

13 To specify how the transformation deals with exception conditions in the loading of this lookup table, click the **Exception** tab. Review the default values, and then select and configure responses to various conditions, such as **Lookup Value Not Found**. These entries apply only to the currently selected lookup table.

Note: Responses to error conditions for the entire transformation are specified in the **Error** tab of the Lookup transformation. △

14 Click **OK** to complete the definition of the lookup and load operations for the first lookup table.

15 For each of the other lookup tables, repeat the steps that define the match between source table and lookup table. Then repeat the steps that define how the foreign key values are mapped from the lookup table to the target table.

16 Use the Target Designer to create a target fact table. Drop that table onto **Place Target Table Here**.

17 Double-click the table in the job and open the **Columns** tab. Configure the target columns to match the columns in the work table of the Lookup transformation. Save and close the Properties dialog box.

18 Open the Table Loader and open the **Mapping** tab. Right-click and select **Quick Map** to connect the Lookup transformation's work table to the target. Save and close the Properties dialog box.

19 Open the Lookup transformation and open the **Errors** tab. Error handling is not configured by default. You can specify an error limit and error tables for specified columns.

20 Save and close the job. If the dimension tables have been updated, you're ready to run your fact table loading job.

Generating Retained Keys for an SCD Dimension Table

Problem

You want to assign a single unique number to all of the rows in a dimension table that apply to individual members, and combine that identifier with a datetime column to uniquely identify each row in the table (as the primary key).

Solution

Generate retained key values using the SCD Type 2 Loader transformation.

Tasks

Perform the following steps to configure a retained key in a job that includes the SCD Type 2 Loader:

1 In the target's Properties dialog box, use the **Columns** tab to add and configure columns for the begin datetime value, end datetime value, and the retained key value, all of which are numeric.

2 In the **Keys** tab, identify the begin datetime column and the retained key column as the primary key columns.

3 In the Properties dialog box of the SCD Type 2 Loader, configure the **Change Tracking** tab, as described in "Loading a Dimension Table Using Begin and End Datetime Values" on page 349.

4 Open the **Business Key** tab and specify the column or columns that make up the business key.

If the business key includes more than one column, and if only one of those columns identifies members in the source table, order your business key specifications so that the top entry in the list is the column that differentiates between members.

If multiple columns in the source uniquely identify members, the column order in the **Business Key** tab is not significant. In this case, you'll need to select a check box in the **Generated Key** tab, as described in a subsequent step.

5 Open the **`Generated Key`** tab and specify the column you defined in the target that will receive the retained key values.

6 Select the check box **`Generate retained key`**. As you do so, note that the field **`Changed record`** becomes unavailable. This happens because changed records receive the same key value that was present in the original record.

7 If you have a multi-column business key, and if more than one of those columns is used to uniquely identify members in the source, then select the check box **`Generate unique keys for each column in the business key`**. Selecting this check box ensures that all columns of the business key are evaluated to determine whether a source row represents a new member, an updated member, or no change to an existing member. Click **`OK`** and open the target's Properties dialog box.

8 In the target's Properties dialog box, open the **`Keys`** tab and specify the datetime column and the retained key column as the primary key. Click **`Apply`**.

9 To enhance performance, open the **`Indexes`** tab to create a composite index of the datetime column and the retained key column. Remove any other columns from the index.

10 For each indexed column, select at minimum the **`Unique values`** check box, and consider selecting **`No missing values`**. Click **`Save`** to complete the job.

Updating Closed-Out Rows in SCD Dimension Tables

Problem

You want to update data in closed-out rows of dimension tables that use begin and end datetime values for change tracking.

Solution

Closed-out rows are automatically updated if the changes are not detected between the source row and the current row for that member, and if the end datetime value of the closed-out row is older than the end datetime value of the current row.

Tasks

Perform the following steps to ensure that your SCD dimension table is configured to automatically load updates to closed-out rows:

1 Open the job that loads the SCD dimension table, or create a job as described in "Loading a Dimension Table Using Begin and End Datetime Values" on page 349.

Note: Make sure that your begin and end datetime columns are mapped into target columns. △

2 Open the SCD Type 2 Loader transformation in the job.

3 Open the **`Change Tracking`** tab.

4 Verify that change tracking is implemented with begin and end datetime values.

5 If you are generating datetime values in the transformation, examine the expressions for the begin datetime and end datetime columns.

For the current row, in the end datetime column, specify an expression that generates a future date that far exceeds any value in the end datetime column for closed-out rows. This ensures that the end datetime for the current row will always be later than any end datetime value in a closed-out row.

6 If you are loading datetime values from the source table, examine that data to ensure that the current-row end datetime values are later than the end datetime value for the closed-out rows that will be updated in the dimension table.

7 If you need to load date values from the source, instead of datetime values, note that you have the option of converting the date to a datetime value. This is the default mode for the SCD Type 2 Loader. To confirm that you are using the default mode, open the **`Options`** tab in the Properties dialog box of the SCD Type 2 Loader. Confirm that the value of **`Format Type for Date`** is **`DATE`**.

8 Save changes, close the job, and submit the job.

Optimizing SQL Pass-Through in the SCD Type 2 Loader

Problem

You want to optimize SQL pass-through performance in your SCD Type 2 Loader transformation.

Solution

Specify that your target table use the appropriate database engine (ORACLE, DB2, TERADATA, or ODBC), rather that the SAS or SPDS engine, which improves performance for SQL pass-through.

Tasks

Perform the following steps to ensure that your dimension table uses the appropriate database engine for SQL pass-through:

1 If you are creating a new target table with the Target Designer, specify the appropriate database engine in the DBMS field, which appears in the second panel, as shown in the following diagram.

Display 19.6 Specifying a Database Engine in the DBMS Field

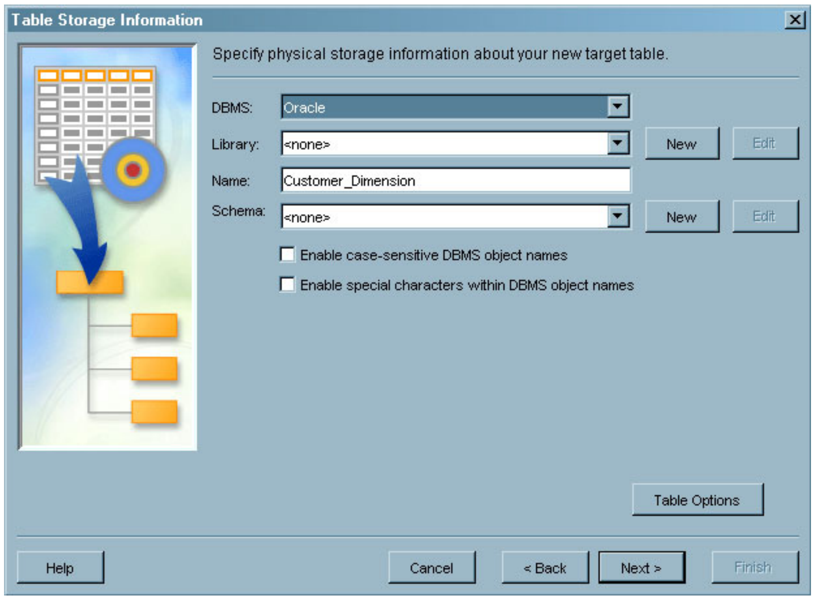


Table Storage Information
Specify physical storage information about your new target table.
DBMS: Oracle
Library: <none>
New
Edit
Name: Customer_Dimension
Schema: <none>
New
Edit
Enable case-sensitive DBMS object names
Enable special characters within DBMS object names
Table Options
Help
Cancel
< Back
Next >
Finish

2 If your target table already exists, right-click the table in the Inventory tree or job and select Properties.

3 In the Properties dialog box, open the **`Physical Storage`** tab.

4 In the **`DBMS`** field, select the appropriate database engine for SQL pass-through. Pass-through is optimized only in the following engines: ORACLE, DB2, TERADATA, and ODBC.

5 Click **`OK`** to save changes and close the dialog box.

Notes:

- SQL pass-through is enabled by default in the **`Options`** tab of the SCD Type 2 Loader, in the field **`Use SQL pass-through`**.
- In the **`Options`** tab, you can set options on the SQL procedure using the **`SQL Options`** field.

CHAPTER 20 Working with Message Queues

About Message Queues

A message queue is a guaranteed message delivery mechanism for handling data sharing in a user-defined format. Several widely used messaging technologies are currently available. The format of the message content can be completely user defined, or it can be a format that has been commonly accepted for a particular industry segment. The message queues in SAS Data Integration Studio support all of the following data transfer types:

Table 20.1 Support Data Transfer Types

Data Transfer Type	Description
Text	Transmits text of a maximum length of 32767 characters or a macro variable for transfer to the message queue. The default value for the **`Text`** field is the ***`etls_qms`*** macro variable. The text is entered directly into the Text field on the **`Queue Options`** tab on the properties windows for the Message Queue Reader and Message Queue Writer transformations.
Tables	Transmits records from a table (from a SAS data set, a DBMS table, or an XML table). In order to successfully handle tables, the structure of the table must be included on the receiving end so that input data values can be correctly formatted to accurately reconstitute the data. A queue is mapped to the data set or table. Each message sent to the queue corresponds to a database record.
Binary Files	Transmits files, provided that the receiver understands the file format.

Unlike other SAS Data Integration Studio jobs, message queue jobs can handle both structured data such as tables and unstructured data such as texts.

Prerequisites for Message Queues

The following prerequisites are required to use message queues in SAS Data Integration Studio jobs:

- Base SAS and SAS Integration technologies must be installed on the machine where the message queue server is installed.
- The message queue server must be installed (WebSphere MQ server for WebSphere MQ queues; MSMQ Server for Microsoft MQ queues). Then, the queues must be defined on the server.
- The workspace server must have client/server or client access to the message queue server. The workspace server defined and used to run queue jobs is critical. For example, if you are using a metadata server on your machine and using the workspace server on Machine X and the model is client/server, then messages are sent to the message queue server running on Machine X.
- The machine used to run the job is able to access the message queue server.
- The queue manager and queues must be defined in SAS Management Console. For more information, see the Administering Message Queues section in the "Administering SAS Data Integration Studio" chapter of the SAS *Intelligence Platform: System Administration Guide*.

Selecting Message Queue Transformations

Problem

You want to select the transformations that are appropriate for a Microsoft or WebSphere message queue that contains information that you need to either send or receive.

Solution

Four transformations are provided in SAS Data Integration Studio to facilitate the processing of message queues. Select the transformations that you need for your process from the table in the Tasks section.

Tasks

Table 20.2 Message Queue Transformations

Transformation	**Purpose**
Microsoft Queue Writer transformation	Enables writing files in binary mode, tables, or structured lines of text to the Microsoft MQ messaging system. The queue and queue manager objects necessary to get to the messaging system are defined in SAS Management Console.
WebSphere Queue Writer transformation	Enables writing files in binary mode, tables, or structured lines of text to the WebSphere MQ messaging system. The queue and queue manager objects necessary to get to the messaging system are defined in SAS Management Console.

Transformation	Purpose
Microsoft Queue Reader transformation	Enables content from a Microsoft MQ message queue to be delivered to SAS Data Integration Studio. If the message is being sent into a table, the message queue content is sent to a table or a Data Integration Studio transformation. If the message is being sent to a macro variable or file, then these files or macro variables can be referenced by a later step.
WebSphere Queue Reader transformation	Enables content from a WebSphere MQ message queue to be delivered to SAS Data Integration Studio. If the message is being sent into a table, the message queue content is sent to a table or a Data Integration Studio transformation. If the message is being sent to a macro variable or a SAS data set file, then these data set files or macro variables can be referenced by a later step.

Additional Information

For detailed information about the Microsoft Queue Writer transformation, see the "About WebSphere Queue Writer Transformations" topic in the SAS Data Integration Studio help. For information about the WebSphere Queue Writer transformation, see the "About WebSphere Queue Writer Transformations" topic. For information about the Microsoft Queue Reader transformation, see the "About Microsoft Queue Reader Transformations" topic. For information about the WebSphere Queue Reader transformation, see the "About WebSphere Queue Reader Transformations" topic. For detailed information about a specific usage issue, see the "SAS Data Integration Studio Usage Notes" topic.

Processing a WebSphere MQ Queue

Problem

You want to write rows from a source table into a message queue. Then, you need to read the messages in the queue from the queue into a target table.

Solution

You can use the WebSphere Queue Writer transformation in SAS Data Integration Studio to write the data to the message queue. Then, you can use the WebSphere Queue Reader transformation to read the message from the queue and populate them into a target table. In addition, you can use the Message browser window in the IBM WebSphere MQ Explorer application to browse the messages. This enables you to monitor the flow of data from the source table to the message queue and from the message queue to the target table.

Text and file transfers are also supported in message queues, but these transfers are not covered in this example. Finally, SAS Data Integration Studio can also process Microsoft MQ queues. However, this example focuses exclusively on the WebSphere queues because they are more commonly encountered.

Tasks

Create the WebSphere Queue Writer Job

Perform the following steps to create and populate the job:

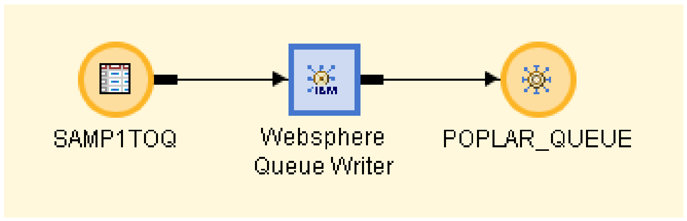
1 Create an empty job.

2 Select and drag the WebSphere Queue Writer transformation from the **`Access`** folder in the Process Library tree into the empty job in the **`Process Editor`** tab in the Process Designer window.

3 Drop the source table on the source drop zone for the WebSphere Queue Writer transformation.

4 Drop the target table on the target drop zone for the WebSphere Queue Writer transformation. The job resembles the following sample.

Display 20.1 Write Records from Table to Queue Job



Configure and Run the WebSphere Queue Writer Job

Perform the following steps to configure the job:

1 Open the **`Queue Options`** tab of the properties window for the WebSphere Queue Writer transformation.

2 Select **`Table`** in the **`Message Type`** group box. Save the setting and close the properties window.

3 Open the IBM WebSphere MQ application to verify that the message queue is empty. Open **`IBM WebSphere MQ`** in the **`Navigator`** section of the application. Navigate to the **`Queues`** item under the **`Queue Managers`** and **`POPLAR_QMGR`** folders. Double-click **`Queues`** to access the Queues table.

4 Right-click the row for **`POPLAR_QMGR`** and click **`Browse Messages`** to access the Message browser window.

5 Verify that no messages are displayed in the window. This step verifies that the message queue is empty before the message queue reader job is run.

6 Run the job. If you are prompted to do so, enter a user ID and password for the default SAS Application Server that generates and run SAS code for the job. The server executes the SAS code for the job.

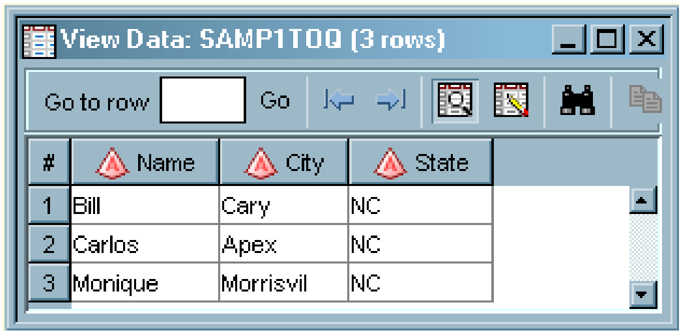
7 If the job completes without error, go to the next section. If error messages appear, read and respond to the messages.

Verify the WebSphere Queue Writer Job

Perform the following steps to verify the results of the queue writer job:

1 Access the View Data window for the source table. A sample source table is shown in the following example.

Display 20.2 Sample Source Table Data


View Data: SAMP1TOQ (3 rows)

#	Name	City	State
1	Bill	Cary	NC
2	Carlos	Apex	NC
3	Monique	Morrisvil	NC

2 Click **`Refresh`** on the Browse messages window in IBM WebSphere MQ. The messages that were written to the sample queue are displayed, and can be used to ensure that the data is consistent.

If the Message data column is not populated, double-click a row in the table to access the Messages Properties window. You can enable the Message data column on the **`Data`** tab.

3 Notice that the rows in the Message Data column of the Browse messages window reflect the corresponding rows in the source table. If you do not see the data that you expected, check the Message Format column on the **`Columns`** tab in the WebSphere Queue Writer Properties window. To access this window, right-click WebSphere Queue Writer in the Process Flow Diagram, and click **`Properties`**. You can compare this column to message and formatting columns in the Browse messages window. Then, you can correct the formats as needed.

Create the WebSphere Queue Reader Job

Perform the following steps to create the WebSphere Queue Reader Job:

1 Create an empty job.

2 Select and drag the WebSphere Queue Reader transformation from the *Access* folder in the Process Library tree into the empty job in the **`Process Editor`** tab in the Process Designer window.

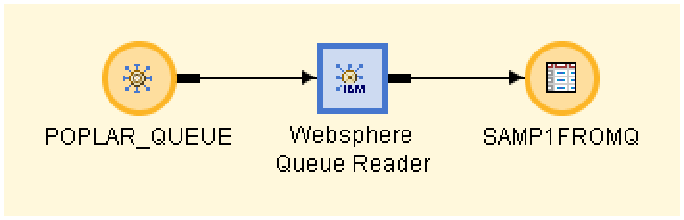
3 Drop the source table on the source drop zone for the WebSphere Queue Reader transformation.

4 Drop the target table on the target drop zone for the WebSphere Queue Reader transformation.

5 Delete the Table Loader and the temporary worktable from the job.

6 After these steps have been completed, the process flow diagram for this example resembles the following display.

Display 20.3 Read Records to a Table Job



Configure and Run the WebSphere Queue Reader Job

Perform the following steps to configure the job:

1 Open the **`Queue Options`** tab of the properties window for the WebSphere Queue Writer transformation.

2 Select **`Table`** in the **`Message Type`** group box. Save the setting and close the properties window. Remember that you verified that the message queue contained the messages from the source table in the Verify the WebSphere Queue Writer Job section above.

3 Run the job. If you are prompted to do so, enter a user ID and password for the default SAS Application Server that generates and run SAS code for the job. The server executes the SAS code for the job.

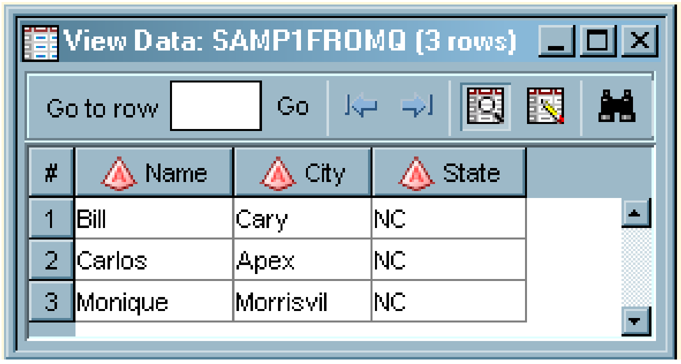
4 If the job completes without error, go to the next section. If error messages appear, read and respond to the messages.

Verify the WebSphere Queue Reader Job

Perform the following steps to verify the results of the queue reader job:

1 Access the View Data window for the source table. A sample source table is shown in the following example.

Display 20.4 Sample Target Table Data



#	Name	City	State
1	Bill	Cary	NC
2	Carlos	Apex	NC
3	Monique	Morrisvil	NC

The source table and the target table contain identical data. This means that the data was transferred successfully through the WebSphere message queue. If you do not see the data that you expected, check the Message Format column on the **`Columns`** tab in the WebSphere Queue Writer Properties window. To access this window, right-click WebSphere Queue Writer in the Process Flow Diagram, and click **`Properties`**. You can compare this information to the Message Browser and the source table. Then, you can correct the formats as needed.

2 Click **`Refresh`** on the Browse messages window in IBM WebSphere MQ. Notice that the Browse messages window is empty. The queue has been cleared until data is written to it in a later job.

Additional Information

For information about processing a Microsoft MQ message queue, see the "Example: Process a Microsoft MQ Queue" topic in SAS Data Integration Studio help.

CHAPTER 21
Working with SPD Server Cluster Tables

About SPD Server Clusters

The SAS Scalable Performance Data (SPD) Server enables you to create dynamic cluster tables. A dynamic cluster table is two or more SPD Server tables that are virtually concatenated into a single entity, using metadata that is managed by the SPD Server. Dynamic cluster tables can be used as the inputs or outputs in SAS Data Integration Studio jobs.

Before you can create an SPD Server, the following prerequisites must be satisfied:

- Administrators must have installed, started, and registered an SPD Server. The application server that executes the cluster table job must be able to access the SPD Server. For more information about SPD Servers, see the chapters about common data sources in the *SAS Intelligence Platform: Data Administration Guide*.
- An SPD Server library must be available. For more information about SPD Server libraries, see the chapters about common data sources in the *SAS Intelligence Platform: Data Administration Guide*.
- A cluster table has been registered in the SPD Server library. For more information, see “Registering Tables with the Target Table Wizard” on page 87.
- All of the tables that are to be added to the cluster table have been registered in the SPD Server library. Each table must also have the same column structure as the cluster table.

Creating an SPD Server Cluster

Problem

You want to create an SPD Server cluster. The cluster table and the tables that you will include in the cluster must be registered in the same SPD Server library and share a common column structure. These cluster tables can be used as the inputs or outputs in SAS Data Integration Studio jobs and can improve the performance of the jobs.

Solution

You can use the Create or Add to a Cluster transformation to create or add tables to an SPD Server cluster table. Use this transformation to create an SPD Server cluster table in a SAS Data Integration Studio job and list its contents in the **`Output`** tab in the Process Designer window.

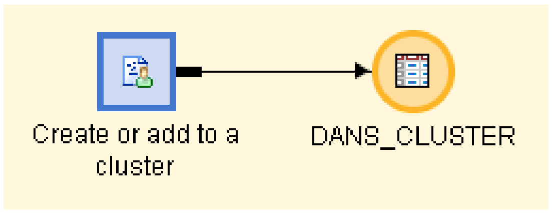
Tasks

Build the SPD Server Cluster

Perform the following steps to build an SPD Server cluster:

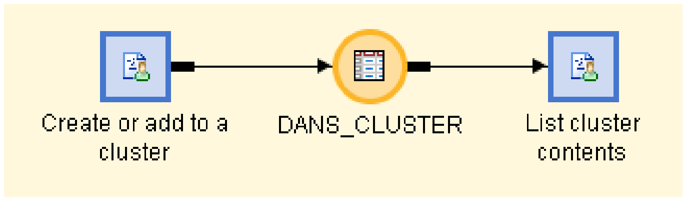
1. Create a job in SAS Data Integration Studio and give it an appropriate name.
2. Drop the Create or Add to a Cluster transformation on the Process Designer window and then drop an SPD Server cluster table in the SPD Server cluster table drop zone. The Table Loader transformation is populated into the Process Designer window. However, this SPD server cluster job does not actually load a physical table. Instead, it creates a virtual table that combines all of the data from the tables included in the SPD Server library into a virtual table that is processed as a single unit. You must therefore remove the table loader for the job to run successfully. See the following example.

Display 21.1 Sample SPD Server Cluster Table Job with Table Loader Removed



3. You can drop the List Cluster Contents transformation on the Process Designer window and then drag the SPD Server cluster table into the drop zone for the List Cluster Contents transformation. This step creates a single process flow diagram for the job, which is shown in the following example.

Display 21.2 Sample SPD Server Cluster Table Job with List Cluster Contents



The List Cluster Contents transformation sends a list of all tables included in the cluster table to the `Output` tab.

4 Right-click the Create or Add to a Cluster transformation and click **`Properties`** to access the Create or add to a cluster Properties window. Then click **`Options`** to access the **`Options`** tab.

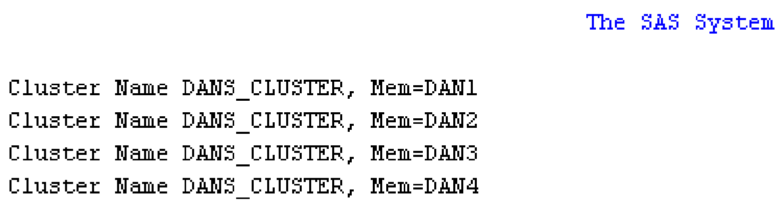
5 Limit the tables included in the cluster table by entering a string in the **`Filter: table name contains ... (Optional)`** field. In this case, enter **`DAN`** because all tables required include this string in the table name.

6 Enter a value into the **`Set maximum number of slots (Optional)`** field. This value must be large enough to accommodate the potential growth of the cluster because the number of slots cannot be increased after the cluster is created. You are required to delete the existing cluster definition and define a new cluster that includes an adequate value for the maximum number of slots.

7 Click **`OK`** to save the setting and close the properties window.

8 Submit and run the job. Click **`Output`** to access the **`Output`** tab and verify that the expected tables were added to the SPD Server cluster table, as shown in the following example:

Display 21.3 Cluster Contents on Output Tab



```
                                  The SAS System

Cluster Name DANS_CLUSTER, Mem=DAN1
Cluster Name DANS_CLUSTER, Mem=DAN2
Cluster Name DANS_CLUSTER, Mem=DAN3
Cluster Name DANS_CLUSTER, Mem=DAN4
```

9 Save the job and check in the repository for future use.

Maintaining an SPD Server Cluster

Problem

You want to maintain an existing SPD server cluster by generating a list of tables included in a cluster or removing a cluster definition.

Solution

You can use the List Cluster Contents transformation or the Remove Cluster transformation. These transformations are explained in the following table.

Table 21.1 SPD Server Transformations

Server	**Tasks That Require This Server**
Generate a list of tables in a cluster	Perform the following steps to use the List Cluster Contents transformation: **1** Create an empty job. **2** Drop the List Cluster Contents transformation into the Process Designer window. **3** Drop the cluster table into the drop zone for the List Cluster Contents transformation. **4** Run the job. Note that you can also include the List Cluster Contents transformation in an SPD server cluster job. Then, you will generate a cluster list each time you create a cluster.
Remove a cluster definition	Perform the following steps to use the Remove Cluster transformation: **1** Create an empty job. **2** Drop the Remove Cluster transformation into the Process Designer window. **3** Drop the cluster table into the drop zone for the List Cluster Contents transformation. **4** Run the job.

PART 4

Appendixes

APPENDIX 1 Recommended Reading

Recommended Reading

Here is the recommended reading list for this title:

- *Customer Data Integration: Reaching a Single Version of the Truth*
- *Cody's Data Cleaning Techniques Using SAS Software*
- *Communications Access Methods for SAS/CONNECT and SAS/SHARE*
- *Moving and Accessing SAS Files*
- *PROC SQL: Beyond the Basics Using SAS*
- *SAS Intelligence Platform: Application Server Administration Guide*
- *SAS Intelligence Platform: Desktop Application Administration Guide*
- *SAS Intelligence Platform: Data Administration Guide*
- *SAS Intelligence Platform: Installation Guide*
- *SAS Intelligence Platform: Overview*
- *SAS Intelligence Platform: Security Administration Guide*
- *SAS Intelligence Platform: System Administration Guide*
- *SAS Management Console: User's Guide*
- *SAS OLAP Server: Administrator's Guide*
- *SAS SQL Procedure User's Guide*

For a complete list of SAS publications, see the current *SAS Publishing Catalog*. To order the most current publications or to receive a free copy of the catalog, contact a SAS representative at

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513
Telephone: (800) 727-3228*
Fax: (919) 677-8166
E-mail: **sasbook@sas.com**
Web address: **support.sas.com/pubs**
* For other SAS Institute business, call (919) 677-8000.

Customers outside the United States should contact their local SAS office.

Glossary

administrator
the person who is responsible for maintaining the technical attributes of an object such as a table or a library. For example, an administrator might specify where a table is stored and who can access the table. See also owner.

alternate key
another term for unique key. See unique key.

analysis data set
in SAS data quality, a SAS output data set that provides information on the degree of divergence in specified character values.

business key
one or more columns in a dimension table that comprise the primary key in a source table in an operational system.

change analysis
the process of comparing one set of metadata to another set of metadata and identifying the differences between the two sets of metadata. For example, in SAS Data Integration Studio, you have the option of performing change analysis on imported metadata. Imported metadata is compared to existing metadata. You can view any changes in the Differences window and choose which changes to apply. To help you understand the impact of a given change, you can run impact analysis or reverse impact analysis on tables and columns in the Differences window.

change management
in the SAS Open Metadata Architecture, a facility for metadata source control, metadata promotion, and metadata replication.

change-managed repository
in the SAS Open Metadata Architecture, a metadata repository that is under metadata source control.

cluster
in SAS data quality, a set of character values that have the same match code.

comparison result
the output of change analysis. For example, in SAS Data Integration Studio, the metadata for a comparison result can be selected, and the results of that comparison can be viewed in a Differences window and applied to a metadata repository. See also change analysis.

cross-reference table
a table that contains only the current rows of a larger dimension table. Columns generally include all business key columns and a digest column. The business key column is used to determine if source rows are new dimensions or updates to existing dimensions. The digest column is used to detect changes in source rows that might update an existing dimension. During updates of the fact table that is associated with the dimension table, the cross-reference table can provide generated keys that replace the business key in new fact table rows.

custom repository
in the SAS Open Metadata Architecture, a metadata repository that must be dependent on a foundation repository or custom repository, thus allowing access to metadata definitions in the repository or repositories on which it depends. A custom repository is used to specify resources that are unique to a particular data collection. For example, a custom repository could define sources and targets that are unique to a particular data warehouse. The custom repository would access user definitions, group definitions, and most server metadata from the foundation repository. See also foundation repository, project repository.

data analysis
in SAS data quality, the process of evaluating input data sets in order to determine whether data cleansing is needed.

data cleansing
the process of eliminating inaccuracies, irregularities, and discrepancies from data.

data integration
the process of consolidating data from a variety of sources in order to produce a unified view of the data.

data lineage
a search that seeks to identify the tables, columns, and transformations that have an impact on a selected table or column. See also impact analysis, reverse impact analysis, transformation.

data store
a table, view, or file that is registered in a data warehouse environment. Data stores can contain either individual data items or summary data that is derived from the data in a database.

data transformation
in SAS data quality, a cleansing process that applies a scheme to a specified character variable. The scheme creates match codes internally to create clusters. All values in each cluster are then transformed to the standardization value that is specified in the scheme for each cluster.

database library
a collection of one or more database management system files that are recognized by SAS and that are referenced and stored as a unit. Each file is a member of the library.

database server
a server that provides relational database services to a client. Oracle, DB/2 and Teradata are examples of relational databases.

delimiter
a character that separates words or phrases in a text string.

derived mapping
a mapping between a source column and a target column in which the value of the target column is a function of the value of the source column. For example, if two

tables contain a Price column, the value of the target table's Price column might be equal to the value of the source table's Price column multiplied by 0.8.

delivery transport
in the Publishing Framework, the method of delivering a package to the consumer. Supported transports include e-mail, message queue, and WebDAV. Although not a true transport, a channel also functions as a delivery mechanism.

digest column
a column in a cross-reference table that contains a concatenation of encrypted values for specified columns in a target table. If a source row has a digest value that differs from the digest value for that dimension, then changes are detected and the source row becomes the new current row in the target. The old target row is closed out and receives a new value in the end date/time column.

dimension
a category of contextual data or detail data that is implemented in a data model such as a star schema. For example, in a star schema, a dimension named Customers might associate customer data with transaction identifiers and transaction amounts in a fact table.

dimension table
in a star schema or snowflake schema, a table that contains data about a particular dimension. A primary key connects a dimension table to a related fact table. For example, if a dimension table named Customers has a primary key column named Customer ID, then a fact table named Customer Sales might specify the Customer ID column as a foreign key.

dynamic cluster table
two or more SAS SPD Server tables that are virtually concatenated into a single entity, using metadata that is managed by the SAS SPD Server.

fact table
the central table in a star schema or snowflake schema. A fact table typically contains numerical measurements or amounts and is supplemented by contextual information in dimension tables. For example, a fact table might include transaction identifiers and transaction amounts. Dimension tables could add contextual information about customers, products, and salespersons. Fact tables are associated with dimension tables via key columns. Foreign key columns in the fact table contain the same values as the primary key columns in the dimension tables.

foreign key
one or more columns that are associated with a primary key or unique key in another table. A table can have one or more foreign keys. A foreign key is dependent upon its associated primary or unique key. In other words, a foreign key cannot exist without that primary or unique key.

foundation repository
in the SAS Open Metadata Architecture, a metadata repository that is used to specify metadata for global resources that can be shared by other repositories. For example, a foundation repository is used to store metadata that defines users and groups on the metadata server. Only one foundation repository should be defined on a metadata server. See also custom repository, project repository.

generated key
a column in a dimension table that contains values that are sequentially generated using a specified expression. Generated keys are used to implement surrogate keys and retained keys.

generated transformation
in SAS Data Integration Studio, a transformation that is created with the Transformation Generator wizard, which helps you specify SAS code for the transformation. See also transformation.

global resource
an object, such as a server or a library, that is shared on a network.

impact analysis
a search that seeks to identify the tables, columns, and transformations that would be affected by a change in a selected table or column. See also transformation, data lineage.

intersection table
a table that describes the relationships between two or more tables. For example, an intersection table could describe the many-to-many relationships between a table of users and a table of groups.

iterative job
a job with a control loop in which one or more processes are executed multiple times. Iterative jobs can be executed in parallel. See also job.

iterative processing
a method of processing in which a control loop executes one or more processes multiple times.

job
a collection of SAS tasks that create output.

locale
a value that reflects the language, local conventions, and culture for a geographic region. Local conventions can include specific formatting rules for dates, times, and numbers, and a currency symbol for the country or region. Collating sequences, paper sizes, and conventions for postal addresses and telephone numbers are also typically specified for each locale. Some examples of locale values are French_Canada, Portuguese_Brazil, and Chinese_Singapore.

lookup standardization
a process that applies a scheme to a data set for the purpose of data analysis or data cleansing.

match code
an encoded version of a character value that is created as a basis for data analysis and data cleansing. Match codes are used to cluster and compare character values. See also sensitivity.

message queue
in application messaging, a place where one program can send messages that will be retrieved by another program. The two programs communicate asynchronously. Neither program needs to know the location of the other program nor whether the other program is running. See also delivery transport.

metadata administrator
a person who defines the metadata for servers, metadata repositories, users, and other global resources.

metadata model
a definition of the metadata for a set of objects. The model describes the attributes for each object, as well as the relationships between objects within the model.

metadata object
a set of attributes that describe a table, a server, a user, or another resource on a network. The specific attributes that a metadata object includes vary depending on which metadata model is being used.

metadata repository
a collection of related metadata objects, such as the metadata for a set of tables and columns that are maintained by an application. A SAS Metadata Repository is an example.

metadata server
a server that provides metadata management services to one or more client applications. A SAS Metadata Server is an example.

metadata source control
in the SAS Open Metadata Architecture, a feature that enables multiple users to work with the same metadata repository at the same time without overwriting each other's changes. See also change management.

operational data
data that is captured by one of more applications in an operational system. For example, an application might capture and manage information about customers, products, or sales. See also operational system.

operational system
one or more applications that capture and manage data for an organization. For example, a business might have a set of applications that manage information about customers, products, and sales.

owner
the person who is responsible for the contents of an object such as a table or a library. See also administrator.

parameterized job
a job that specifies its inputs and outputs as parameters. See also job.

parameterized table
a table whose metadata specifies some attributes as variables rather than as literal values. For example, the input to an iterative job could be a parameterized table whose metadata specifies its physical pathname as a variable. See also iterative job.

primary key
one or more columns that are used to uniquely identify a row in a table. A table can have only one primary key. The column(s) in a primary key cannot contain null values. See also unique key, foreign key.

process flow diagram
a diagram in the Process Editor that specifies the sequence of each source, target, and process in a job. In the diagram, each source, target, and process has its own metadata object. Each process in the diagram is specified by a metadata object called a transformation.

project repository
a repository that must be dependent on a foundation repository or custom repository that will be managed by the Change Management Facility. A project repository is used to isolate changes from a foundation repository or from a custom repository. The project repository enables metadata programmers to check out metadata from a foundation repository or custom repository so that the metadata can be modified and tested in a separate area. Project repositories provide a development/testing environment for customers who want to implement a formal change management scheme. See also custom repository, foundation repository.

Quality Knowledge Base
a collection of locales and other information that is referenced during data analysis and data cleansing. For example, to create match codes for a data set that contains street addresses in Great Britain, you would reference the ADDRESS match definition in the ENGBR locale in the Quality Knowledge Base.

register
to save metadata about an object to a metadata repository. For example, if you register a table, you save metadata about that table to a metadata repository.

retained key
a numeric column in a dimension table that is combined with a begin-date column to make up the primary key. During the update of a dimensional target table, source rows that contain a new business key are added to the target. A key value is generated and added to the retained key column and a date is added to the begin-date column. When a source row has the same business key as a row in the target, the source row is added to the target, including a new begin-date value. The retained key of the new column is copied from the target row.

reverse impact analysis
See data lineage.

SAS Application Server
in the SAS Intelligence Platform, a logical entity that represents the SAS server tier. This logical entity contains specific servers (for example, a SAS Workspace Server and a SAS Stored Process Server) that execute SAS code. A SAS Application Server has relationships with other metadata objects. For example, a SAS library can be assigned to a SAS Application Server. When a client application needs to access that library, the client submits code to the SAS Application Server to which the library is assigned.

SAS Management Console
a Java application that provides a single user interface for performing SAS administrative tasks.

SAS metadata
metadata that is created by SAS software. Metadata that is in SAS Open Metadata Architecture format is one example.

SAS OLAP Server
a SAS server that provides access to multidimensional data. The data is queried using the multidimensional expressions (MDX) language.

SAS Open Metadata Architecture
a general-purpose metadata management facility that provides metadata services to SAS applications. The SAS Open Metadata Architecture enables applications to exchange metadata, which makes it easier for these applications to work together.

SAS task
a logical process that is executed by a SAS session. A task can be a procedure, a DATA step, a window, or a supervisor process.

SAS XML library
a library that uses the SAS XML LIBNAME engine to access an XML file.

SAS/CONNECT server
a server that provides SAS/CONNECT services to a client. When SAS Data Integration Studio generates code for a job, it uses SAS/CONNECT software to submit code to remote computers. SAS Data Integration Studio can also use SAS/ CONNECT software for interactive access to remote libraries.

SAS/SHARE library
a SAS library for which input and output requests are controlled and executed by a SAS/SHARE server.

SAS/SHARE server
the result of an execution of the SERVER procedure, which is part of SAS/SHARE software. A server runs in a separate SAS session that services users' SAS sessions by controlling and executing input and output requests to one or more libraries.

SAS Stored Process Server
a SAS IOM server that is launched in order to fulfill client requests for SAS Stored Processes.

scheme
a lookup table or data set of character variables that contains variations of data items and specifies the preferred variation form or standard. When these schemes are applied to the data, the data is transformed or analyzed according to the predefined rules to produce standardized values.

sensitivity
in SAS data quality, a value that specifies the degree of complexity in newly created match codes. A higher sensitivity value results in greater match code complexity, which in turn results in a larger number of clusters, with fewer members in each cluster.

server administrator
a person who installs and maintains server hardware or software. See also metadata administrator.

server component
in SAS Management Console, a metadata object that specifies information about how to connect to a particular kind of SAS server on a particular computer.

slowly changing dimensions
a technique for tracking changes to dimension table values in order to analyze trends. For example, a dimension table named Customers might have columns for Customer ID, Home Address, Age, and Income. Each time the address or income changes for a customer, a new row could be created for that customer in the dimension table, and the old row could be retained. This historical record of changes could be combined with purchasing information to forecast buying trends and to direct customer marketing campaigns.

snowflake schema
tables in a database in which a single fact table is connected to multiple dimension tables. The dimension tables are structured to minimize update anomalies and to address single themes. This structure is visually represented in a snowflake pattern. See also star schema.

source
an input to an operation.

star schema
tables in a database in which a single fact table is connected to multiple dimension tables. This is visually represented in a star pattern. SAS OLAP cubes can be created from a star schema.

surrogate key
a numeric column in a dimension table that is the primary key of that table. The surrogate key column contains unique integer values that are generated sequentially when rows are added and updated. In the associated fact table, the surrogate key is included as a foreign key in order to connect to specific dimensions.

target
an output of an operation.

transformation
a SAS task that extracts data, transforms data, or loads data into data stores.

transformation template
a process flow diagram that consists of a transformation object and one or more drop zones for sources, targets, or both.

unique key
one or more columns that can be used to uniquely identify a row in a table. A table can have one or more unique keys. Unlike a primary key, a unique key can contain null values. See also primary key, foreign key.

Web service
a programming interface that enables distributed applications to communicate even if the applications are written in different programming languages or are running on different operating systems.

Index

Your Turn