



THE  
POWER  
TO KNOW.

# **SAS<sup>®</sup>** **Data Integration Studio 4.2** **User's Guide**



The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2009. *SAS® Data Integration Studio 4.2: User's Guide*. Cary, NC: SAS Institute Inc.

**SAS® Data Integration Studio 4.2: User's Guide**

Copyright © 2009, SAS Institute Inc., Cary, NC, USA

ISBN 978-1-59047-960-5

All rights reserved. Produced in the United States of America.

**For a hard-copy book:** No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

**For a Web download or e-book:** Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

**U.S. Government Restricted Rights Notice.** Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

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

1st electronic book, February 2009

1st printing, March 2009

2nd electronic book, May 2009

2nd printing, June 2009

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at [support.sas.com/publishing](http://support.sas.com/publishing) or call 1-800-727-3228.

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

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

---

# Contents

## PART 1 Introduction 1

<b>Chapter 1 • Overview of SAS Data Integration</b>	<b>3</b>
About SAS Data Integration	3
Advantages of SAS Data Integration	4
A Basic Data Integration Environment	5
Online Help for SAS Data Integration Studio	8
Administrative Documentation for SAS Data Integration Studio	9
Accessibility Features in SAS Data Integration Studio	10
Upgrading from Earlier Versions	13

## PART 2 General User Tasks 19

<b>Chapter 2 • Getting Started</b>	<b>21</b>
Setup for SAS Data Integration Studio	22
Security for SAS Data Integration Studio	22
Main Tasks for Creating Process Flows	23
Starting SAS Data Integration Studio	24
Connecting to a SAS Metadata Server	26
Working with the Folders Tree	27
Selecting a Default SAS Application Server	30
Registering SAS Libraries	31
Working with User-Defined Formats	32
Registering Tables and Cubes	33
Specifying Global Options in SAS Data Integration Studio	35
Working with Change Management	36
Add a Note or Document to a Registered Object	41
View the Content of Notes or Documents	43
<b>Chapter 3 • Importing, Exporting, and Copying Metadata</b>	<b>45</b>
Metadata Import and Export in SAS Data Integration Studio	46
Working with SAS Package Metadata	46
Preparing to Import or Export SAS Package Metadata	47
Exporting SAS Package Metadata	48
Importing SAS Package Metadata	49
Copying and Pasting Metadata Objects	51
Working with SAS Metadata Bridges	51
Usage Notes for Importing or Exporting with a SAS Metadata Bridge	52
Preparing to Import or Export with a SAS Metadata Bridge	53
Importing New Metadata with a SAS Metadata Bridge	53
Importing Updated Metadata with a SAS Metadata Bridge	55
Exporting Metadata with a SAS Metadata Bridge	60
<b>Chapter 4 • Working with Tables</b>	<b>63</b>
About Tables	64
Registering Existing Tables with the Register Tables Wizard	65
Registering New Tables with the New Table Wizard	66

Viewing or Updating Table Metadata . . . . .	68
Using a Physical Table to Update Table Metadata . . . . .	69
Specifying Options for Tables . . . . .	70
Supporting Case and Special Characters in Table and Column Names . . . . .	72
Maintaining Column Metadata . . . . .	77
Maintaining Keys . . . . .	82
Maintaining Indexes . . . . .	87
Browsing Table Data . . . . .	89
Editing SAS Table Data . . . . .	92
Using the View Data Window to Create a SAS Table . . . . .	95
Specifying Browse and Edit Options for Tables and External Files . . . . .	96
<b>Chapter 5 • Working with External Files . . . . .</b>	<b>99</b>
About External Files . . . . .	100
Registering a Delimited External File . . . . .	100
Registering a Fixed-Width External File . . . . .	103
Registering an External File with User-Written Code . . . . .	108
Viewing or Updating External File Metadata . . . . .	111
Overriding the Code Generated by the External File Wizards . . . . .	112
Specifying NLS Support for External Files . . . . .	113
Accessing an External File with an FTP Server or an HTTP Server . . . . .	113
Viewing Data in External Files . . . . .	114
Registering a COBOL Data File That Uses a COBOL Copybook . . . . .	115
Using an External File in the Process Flow for a Job . . . . .	117
<b>Chapter 6 • Creating Jobs . . . . .</b>	<b>121</b>
About Jobs . . . . .	122
Creating an Empty Job . . . . .	123
Creating a Process Flow for a Job . . . . .	124
Creating a Job That Contains Jobs . . . . .	125
Working with Default Temporary Output Tables . . . . .	126
About Job Options . . . . .	130
Documenting Process Flow Diagrams . . . . .	133
Accessing Local and Remote Data . . . . .	133
Viewing or Updating Job Metadata . . . . .	136
Displaying the SAS Code for a Job . . . . .	137
Common Code Generated for a Job . . . . .	138
<b>Chapter 7 • Managing Jobs . . . . .</b>	<b>141</b>
About Managing Jobs . . . . .	142
Submitting a Job for Immediate Execution . . . . .	142
Meeting Prerequisites for Collecting Job Statistics . . . . .	145
Reviewing a Successful Job . . . . .	145
Diagnosing and Correcting an Unsuccessful Job . . . . .	150
Maintaining Column Mappings . . . . .	154
Managing the Scope of Column Changes in Jobs . . . . .	158
Managing Connections in Job Editor Windows . . . . .	162
Viewing the Code for a Transformation . . . . .	164
Viewing or Updating the Metadata for Transformations . . . . .	165
<b>Chapter 8 • Managing the Status of Jobs and Transformations . . . . .</b>	<b>167</b>
About Status Handling for Jobs and Transformations . . . . .	167
Default Conditions, Actions, and Conditional Action Sets . . . . .	168
Prerequisites for Actions . . . . .	172
Perform Actions Based on the Status of a Job . . . . .	173
Perform Actions Based on the Status of a Transformation . . . . .	175



Macro Variables for Status Handling . . . . .	177
<b>Chapter 9 • Deploying Jobs . . . . .</b>	<b>183</b>
About Deploying Jobs . . . . .	184
About Deploying Jobs for Scheduling . . . . .	185
Prerequisites for Deploying a Job for Scheduling . . . . .	185
Deploying Jobs for Scheduling . . . . .	185
Redeploying Jobs for Scheduling . . . . .	187
Using Scheduling to Handle Complex Process Flows . . . . .	187
Using Deploy for Scheduling to Execute Jobs on a Remote Host . . . . .	188
About Deploying Jobs as Stored Processes . . . . .	189
Prerequisites for Deploying a Job as a Stored Process . . . . .	190
Deploying Jobs as Stored Processes . . . . .	190
Redeploying Jobs to Stored Processes . . . . .	192
Viewing or Updating Stored Process Metadata . . . . .	193
About Deploying Jobs as Web Services . . . . .	194
Prerequisites for Web Service Jobs . . . . .	195
Requirements for Web Service Jobs . . . . .	195
Creating a Web Service Job . . . . .	196
Deploying a Web Service Job as a Stored Process . . . . .	200
Deploying a Stored Process as a Web Service . . . . .	203
<b>Chapter 10 • Working with Generated Code . . . . .</b>	<b>205</b>
About Code Generated for Jobs . . . . .	205
Displaying the Code Generated for a Job . . . . .	209
Displaying the Code Generated for a Transformation . . . . .	209
Specifying Options for Jobs . . . . .	210
Specifying Options for a Transformation . . . . .	210
Modifying Configuration Files or SAS Start Commands for Application Servers . . . . .	211
<b>Chapter 11 • Working with User-Written Code . . . . .</b>	<b>213</b>
About User-Written Code . . . . .	213
Adding User-Written Code to the Precode and Postcode Tab . . . . .	214
Adding a User Written Code Transformation to a Job . . . . .	216
Creating and Using a Generated Transformation . . . . .	219
Maintaining a Generated Transformation . . . . .	226
Editing the Generated Code for a Job or Transformation . . . . .	228
Replacing the Generated Code for a Job or Transformation . . . . .	229
<b>Chapter 12 • Optimizing Process Flows . . . . .</b>	<b>231</b>
About Process Flow Optimization . . . . .	231
Managing Process Data . . . . .	232
Managing Columns . . . . .	235
Streamlining Process Flow Components . . . . .	237
Using Simple Debugging Techniques . . . . .	238
Using SAS Logs . . . . .	242
Reviewing Temporary Output Tables . . . . .	244
Additional Performance Optimization Information . . . . .	246
<b>Chapter 13 • Using Impact Analysis . . . . .</b>	<b>247</b>
About Impact Analysis and Reverse Impact Analysis . . . . .	247
Performing an Impact Analysis . . . . .	248
Performing Impact Analysis on a Generated Transformation . . . . .	251
Performing Reverse Impact Analysis . . . . .	253
<b>Chapter 14 • Working with Reports . . . . .</b>	<b>255</b>

About Reports . . . . .	256
Opening the Reports Window . . . . .	256
Selecting the Reports Perspective . . . . .	257
Customizing the Tables Report . . . . .	258
Customizing the Job Documentation Report . . . . .	259
Running and Saving a Report . . . . .	260
Saving a Report As a Document Object . . . . .	262
Viewing a Report . . . . .	263
Creating Your Own Report . . . . .	265

## PART 3 Working with Transformations 267

<b>Chapter 15 • Working with Loader Transformations . . . . .</b>	<b>269</b>
About Loader Transformations . . . . .	269
About the SPD Server Table Loader Transformation . . . . .	270
About the Table Loader Transformation . . . . .	270
Setting Table Loader Transformation Options . . . . .	271
Selecting a Load Technique . . . . .	273
Removing Non-Essential Indexes and Constraints during a Load . . . . .	276
Considering a Bulk Load . . . . .	277
<b>Chapter 16 • Working with SAS Sort Transformations . . . . .</b>	<b>279</b>
About Sort Transformations . . . . .	279
Optimizing Sort Performance . . . . .	279
Creating a Table That Contains the Sorted Contents of a Source . . . . .	282
<b>Chapter 17 • Working with SQL Join Transformations . . . . .</b>	<b>285</b>
About SQL Join Transformations . . . . .	287
Using the Designer Window . . . . .	287
Reviewing and Modifying Clauses, Joins, and Tables in an SQL Query . . . . .	288
Understanding Automatic Joins . . . . .	291
Selecting the Join Type . . . . .	294
Adding User-Written SQL Code . . . . .	296
Debugging an SQL Query . . . . .	297
Adding a Column to the Target Table . . . . .	299
Adding a Join to an SQL Query on the Designer Tab . . . . .	299
Creating a Simple SQL Query . . . . .	301
Configuring a SELECT Clause . . . . .	303
Adding a CASE Expression . . . . .	305
Creating or Configuring a WHERE Clause . . . . .	307
Adding a GROUP BY Clause and a HAVING Clause . . . . .	309
Adding an ORDER BY Clause . . . . .	312
Adding Subqueries . . . . .	313
Validating or Submitting an SQL Query . . . . .	318
Joining a Table to Itself . . . . .	319
Using Parameters with an SQL Join . . . . .	320
Constructing a SAS Scalable Performance Data Server Star Join . . . . .	321
Optimizing SQL Processing Performance . . . . .	322
Performing General Data Optimization . . . . .	323
Influencing the Join Algorithm . . . . .	324
Setting the Implicit Property for a Join . . . . .	325
Enabling Pass-Through Processing . . . . .	326
Using Properties Window Options to Optimize SQL Processing Performance . . . . .	328

<b>Chapter 18 • Working with Iterative Jobs and Parallel Processing</b>	<b>333</b>
About Iterative Jobs	333
Creating and Running an Iterative Job	334
Creating a Parameterized Job	337
Creating a Control Table	340
About Parallel Processing	342
Setting Options for Parallel Processing	344
<b>Chapter 19 • Working with Slowly Changing Dimensions</b>	<b>347</b>
About Slowly Changing Dimensions	348
About Dimension Tables	350
About Fact Tables	352
Loading a Dimension Table with Type 1 and 2 Updates	353
Loading a Fact Table Using Dimension Table Lookup	356
Loading a Table and Adding a Surrogate Primary Key	362
Tracking Changes in Source Datetime Values	365
Closing Out Rows in Datetime Change Tracking	367
<b>Chapter 20 • Working with Change Data Capture</b>	<b>369</b>
About the Change Data Capture Transformations	369
About CDC Changed Data Tables	371
About CDC Control Tables	372
Capture Changed Data from Oracle	372
<b>Chapter 21 • Working with Message Queues</b>	<b>379</b>
About Message Queues	379
Prerequisites for Message Queues	380
Selecting Message Queue Transformations	381
Processing a WebSphere Queue	382
Polling a Websphere Message Queue	384
Processing a Microsoft Queue	386
<b>Chapter 22 • Working with SPD Server Cluster Tables</b>	<b>389</b>
About SPD Server Cluster Tables	389
Creating an SPD Server Cluster Table	390
Maintaining an SPD Server Cluster	391

## PART 4 Appendixes 393

<b>Appendix 1 • Main Windows and Wizards</b>	<b>395</b>
Analysis Window	396
Checkouts Tree	397
Code Editor	397
Comparison Results Window	398
Connection Profile Window	399
Desktop	399
Details Pane	401
Expression Builder	402
Folders Tree	403
Inventory Tree	404
Job Editor	407
Properties Windows	408
Reports Window	411
Tools-Options Window	412

Transformations Tree .....	413
Tree View .....	419
View Data Windows .....	421
Wizards .....	423
<b>Appendix 2 • Java Code and Methods for Report Plug-ins .....</b>	<b>427</b>
Example Java Code for a Report Plug-in .....	427
Reporting Interface Methods .....	433
<b>Glossary .....</b>	<b>439</b>
<b>Index .....</b>	<b>449</b>

## ***Part 1***

---

# Introduction

### *Chapter 1*

## ***Overview of SAS Data Integration*** ..... 3



## Chapter 1

# Overview of SAS Data Integration

---

<b>About SAS Data Integration</b> .....	<b>3</b>
<b>Advantages of SAS Data Integration</b> .....	<b>4</b>
<b>A Basic Data Integration Environment</b> .....	<b>5</b>
Overview of a Data Integration Environment .....	5
SAS Management Console .....	5
SAS Data Integration Studio .....	6
Servers .....	6
Libraries .....	8
Additional Information .....	8
<b>Online Help for SAS Data Integration Studio</b> .....	<b>8</b>
<b>Administrative Documentation for SAS Data Integration Studio</b> .....	<b>9</b>
<b>Accessibility Features in SAS Data Integration Studio</b> .....	<b>10</b>
Overview .....	10
Enabling Assistive Technologies .....	10
Accessibility Standards .....	10
<b>Upgrading from Earlier Versions</b> .....	<b>13</b>
Overview .....	13
Objects That Are Not Migrated .....	14
Updates to Jobs and Transformations During Migration .....	14
User Action Required for Migrated Jobs with Data Quality Transformations .....	15
Updates to Jobs and Transformations during Partial Promotion .....	15
Changes to the Tree View .....	16
SAS Workspace Server Requirements for New Jobs .....	16
Impacts on Change Management .....	17
Impacts on SAS Solutions .....	17
Migration Web Site .....	17

---

## 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) system, 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. 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.

---

## 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.

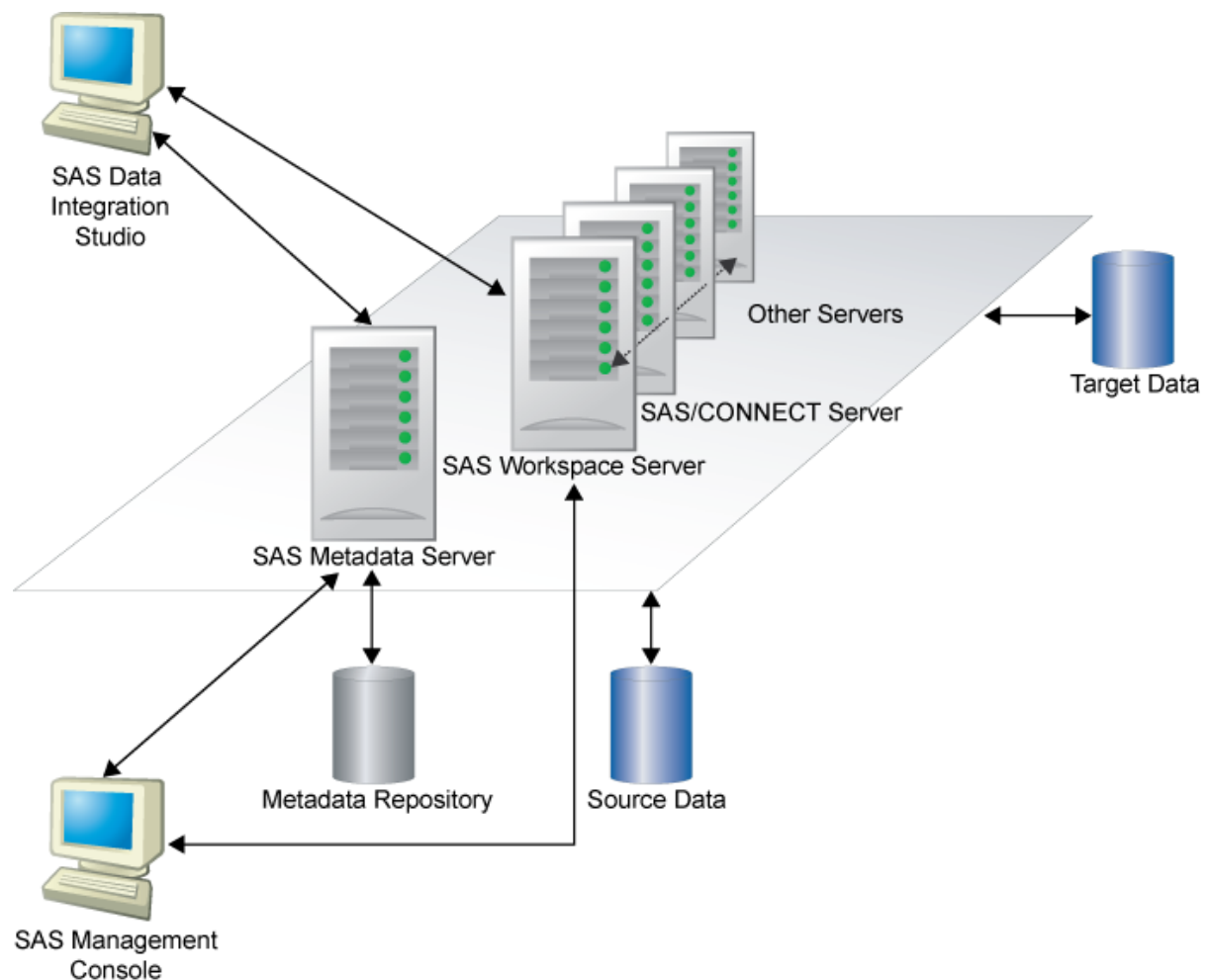


## A Basic Data Integration Environment

### 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 for building, implementing and managing data integration processes regardless of data sources, applications, or platforms. 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 that is generated by SAS Data Integration Studio
- user-written source code

## 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. By default, this application server 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 that is 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 the Server Is Used	How the Server Is Specified
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.

Server	How the Server Is Used	How the Server Is Specified
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 that are 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 the Server Is Used	How the Server Is 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 the Register Tables wizard or the New Tables wizard, 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 can be used to access an Oracle database:

```
libname mydb oracle user=admin1 pass=adlmin
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 (ERM) Systems**

Optional Composite Software provides access to ERM systems such as Siebel, PeopleSoft, Oracle Applications and Salesforce.com. An optional data surveyor wizard provides access to SAP ERM systems. For details about Composite Software and the data surveyor wizard for SAP ERM systems, see the *SAS Intelligence Platform: Data Administration Guide*.

## **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.

## **Additional Information**

For more information about setting up a data integration environment, administrators should see [“Administrative Documentation for SAS Data Integration Studio” on page 9](#).

---

## **Online Help for SAS Data Integration Studio**

The online Help describes all windows in SAS Data Integration Studio, and it includes all topics in the user's guide. 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.
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

Administrative tasks that are performed outside of the SAS Data Integration Studio interface are described in SAS Intelligence Platform documentation, which can be found at the following location: <http://support.sas.com/92administration>.

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
<ul style="list-style-type: none"> <li>Set up a folder structure for your site in the <b>Folders</b> tree.</li> <li>Promote metadata (additional information and metadata export and import).</li> <li>Start, stop, and check the status of servers.</li> <li>Monitor the system and set up system logs.</li> <li>Back up and restore your system.</li> <li>Optimize the performance of the SAS Metadata Server.</li> <li>Manage SAS metadata repositories.</li> </ul>	SAS Intelligence Platform: System Administration Guide
<ul style="list-style-type: none"> <li>Set up security.</li> </ul>	SAS Intelligence Platform: Security Administration Guide
<ul style="list-style-type: none"> <li>Set up data servers and libraries for common data sources.</li> </ul>	SAS Intelligence Platform: Data Administration Guide
<ul style="list-style-type: none"> <li>Set up SAS Application Servers.</li> </ul>	SAS Intelligence Platform: Application Server Administration Guide
<ul style="list-style-type: none"> <li>Set up grid computing (so that jobs can execute on a grid).</li> </ul>	Grid Computing for SAS 9.2
<ul style="list-style-type: none"> <li>Set up scheduling for jobs that have been deployed for scheduling.</li> </ul>	Scheduling In SAS

Administrative Task	Related Documentation
<ul style="list-style-type: none"> <li>Set up change management.</li> <li>Set up servers and libraries for remote data (multi-tier environments).</li> <li>Set up support for message queue jobs.</li> <li>Set up support for Web service jobs and other stored process jobs.</li> <li>Enable the bulk-loading of data into target tables in a DBMS.</li> <li>Set up SAS Data Quality software.</li> <li>Set up support for job status handling.</li> <li>Set up support for FTP and HTTP access to external files.</li> </ul>	SAS Intelligence Platform: Desktop Application Administration Guide
<ul style="list-style-type: none"> <li>Work with SAS OLAP cubes.</li> </ul>	SAS OLAP Server: User's Guide

## Accessibility Features in SAS Data Integration Studio

### Overview

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.

If you have questions or concerns about the accessibility of SAS products, send e-mail to [accessibility@sas.com](mailto:accessibility@sas.com).

### Enabling Assistive Technologies

For instructions about how to configure SAS Data Integration Studio software so that assistive technologies 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*.

### Accessibility Standards

SAS Data Integration Studio follows the standards that are recommended in the *Java Look and Feel Design Guidelines, Second Edition* (available at [java.sun.com](http://java.sun.com)). All known exceptions are documented in the following table. SAS is committed to improving the accessibility and usability of our products. Many of the issues will be addressed within future releases of the application.

Table 1.4 Accessibility Exceptions

Accessibility Issue	Support Status	Explanation
Keyboard equivalents for user actions.	Supported with exceptions	<p>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.</p> <p>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 <b>Diagram</b> tab in the Job Editor window, or the <b>Designer</b> tab in the SQL Join properties window.</p> <p>Specifically, use of the <b>Diagram</b> tab in the Job Editor and the <b>Designer</b> 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 require 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.</p> <p><b>Example:</b> Use of the <b>Diagram</b> tab in the Job Editor 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.</p> <p>The software supports keyboard equivalents to navigating between different prompts in a window. If the TAB key does not move focus to the next prompt, press CTRL+TAB to access the next prompt.</p> <p>When you are defining or editing a static list in a prompt, if pressing SPACEBAR once does not select or clear the check box or radio button, then press SPACEBAR twice to select or clear a default value selection.</p> <p>If focus is transferred to another prompt after you finish editing a row, use the TAB key or SHIFT+TAB until focus is back on the prompt you want, and then you can use the TAB key or the arrow keys to navigate through the rows of values.</p>

Accessibility Issue	Support Status	Explanation
Keyboard equivalents for user actions.	Supported with exceptions	In a window with multiple tabs, sometimes pressing CTRL+TAB can switch to another tab instead of moving to the next prompt in the current tab. If the current prompt exhibits this behavior, press TAB instead of CTRL+TAB to move focus to the next prompt in the current tab. In general, press TAB to move to the next prompt in the current tab, and press only CTRL+TAB if TAB by itself adds space to the current prompt.
Identity, operation, and state of interface elements.	Supported with exceptions	In some wizards, identity, operation, and state of some interface elements is ambiguous. SAS plans to address these issues in a future release.  <b>Example:</b> When you select a library in the Register Tables 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 reads 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.
Application override of user-selected contrast and color selections and other individual display attributes.	Supported with exceptions	SAS Data Integration Studio inherits the color and contrast settings of the operating system with the following exception:  As with most other Java applications, system font settings are not inherited in the main application window. If you need larger fonts, then consider using a screen magnifier.
Color alone as the only significant difference in controls or displays.	Supported with exceptions	In the Authorization dialog box, and on the <b>Authorization</b> tab in the properties windows for some objects, the background colors of the check boxes in the permissions table indicate how a permission is assigned. For information about the meaning of each color, see the Help for the <b>Authorization</b> tab or dialog box.



Accessibility Issue	Support Status	Explanation
Electronic forms and displays.	Supported with exceptions	<p>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 that you press the TAB key and listen closely to the items, or (2) type the path explicitly.</p> <p>When the user sets the operating system settings to high contrast, some attributes of that setting are not inherited. <b>Example:</b> In some wizards such as the Register Tables wizard, 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.</p>
F1 key	SAS plans to address this issue in a future release.	The F1 key does not open the Help for the New Prompt and Edit Prompt dialog boxes. The workaround is to click the <b>Help</b> button at the bottom of dialog boxes.
JAWS reader	SAS plans to address this issue in a future release.	For any window or dialog box that contains a table, JAWS cannot read the column and row headings. JAWS can read the contents of the table cells, but not the headings, so the context might be confusing.
JAWS focus on a list box	SAS plans to address this issue in a future release.	<p>For any Open, Save, or Select dialog box that does not display items in a tree, when the focus is on the list box, JAWS can read the name of the selected item only. If you use the arrow keys to navigate through the list of items, JAWS does not read the names of any of the items that are not selected.</p> <p>To enable JAWS to read the name of an item, select the item in the list box, and then use the TAB key to move back into the list box. After you move back into the list box, JAWS can read the name of the selected item.</p>

---

## Upgrading from Earlier Versions

### Overview

SAS Data Integration Studio users who are migrating to version 4.2 should take special note of the following changes:

- [“Objects That Are Not Migrated” on page 14](#)
- [“Updates to Jobs and Transformations During Migration” on page 14](#)

- “User Action Required for Migrated Jobs with Data Quality Transformations” on page 15
- “Updates to Jobs and Transformations during Partial Promotion” on page 15
- “Changes to the Tree View” on page 16
- “SAS Workspace Server Requirements for New Jobs” on page 16
- “Impacts on Change Management” on page 17
- “Impacts on SAS Solutions” on page 17
- “Migration Web Site” on page 17

### ***Objects That Are Not Migrated***

SAS 9.1 metadata objects for external files, and any jobs that include these objects, will not be migrated. The external files must be re-registered in the 9.2 environment. Any jobs that include SAS 9.1 metadata objects for external files must be recreated in the 9.2 environment.

The Forecasting transformation is not supported in SAS Data Integration Studio 4.2. Jobs that include the Forecasting transformation will not be migrated.

User-defined job status Conditions and Actions are not supported in SAS Data Integration Studio 4.2. Jobs that include custom Conditions and Actions will not be migrated.

### ***Updates to Jobs and Transformations During Migration***

After migration, you will see the following changes to jobs and transformations.

All migrated transformations, including generated transformations before 3.2, are converted to use the new prompting framework. For most users, the main impact of this change is on the **Options** tab for transformations. It is now easier to select options from this tab. After migration, jobs with updated transformations should produce the same output as before.

Any versions of the Table Loader transformation and the SQL Join transformation created before version 3.4 are replaced with the latest versions of these transformations. After migration, jobs with converted transformations should produce the same output as before.

Any job whose process flow included a separate Report transformation, such as Correlations Report and Frequency Report, no longer have that transformation in the process flow. The need for separate Report transformations has been eliminated. After migration, jobs that used to require the Report transformations should produce the same output as before.

All jobs that were deployed for scheduling or were deployed as stored processes should be redeployed in the SAS 9.2 environment. From the SAS Data Integration Studio desktop, you can select **Tools** ⇒ **Redeploy Jobs for Scheduling** or **Tools** ⇒ **Redeploy Jobs to Stored Processes**.

Any job that was deployed as a Web Service in SAS Data Integration Studio 3.4 now appears in the Stored Process folder of the Inventory tree, not the Web service (generated) folder. This special kind of stored process can be used as before.

## **User Action Required for Migrated Jobs with Data Quality Transformations**

If jobs that include the Create Match Code transformation do not run successfully after migration, verify that the appropriate Quality Knowledge Base (QKB) location value (DQSETUPLOC value), is specified on the global options window for SAS Data Integration Studio. To verify the DQSETUPLOC value, select **Tools** ⇒ **Options** from the menu bar, click the **Data Quality** tab, and then verify that the appropriate value is specified in the **DQ Setup Location** field.

Jobs that include the Apply Lookup Standardization transformation will not run successfully after migration until you take one of the following actions:

- Open each affected job and replace the migrated Apply Lookup Standardization transformation with a new (4.2) Apply Lookup Standardization transformation. You must also restore the mappings in each job.
- Alternatively, you can go back to your SAS Data Integration Studio 3.4 environment, export the original Apply Lookup Standardization jobs to SAS 9.1.3 package format, and then use the Import SAS Package wizard in SAS Data Integration Studio 4.2 to convert and register these jobs on your metadata server.

*Note:* The export SAS Package method is possible only if you are using SAS 9.1.3 Service Pack 4, with SAS Data Integration Studio 3.4, and the 34DATABLD09 hotfix installed.

The following additional steps are required in order for the export SAS Package method to work:

1. (Optional) In SAS Data Integration Studio 4.2, if you are not using the migrated jobs that include the Apply Lookup Standardization transformation, delete them. Otherwise, you have to manage multiple copies of the same metadata objects when you import the same jobs as a SAS Package.
2. In SAS Data Integration Studio 3.4, create a package of jobs that use the same scheme repository type in the Apply Lookup Standardization. Create one package for the BFD scheme type and a separate package for the NOBFD scheme type.
3. In SAS Data Integration Studio 4.2, verify that the default DQ Setup Location is correct, and that the default Scheme Repository Type matches the type (BFD or NOBFD) for the package of jobs that you are importing.

To verify these values before importing the SAS Package, select **Tools** ⇒ **Options** from the menu bar, click the **Data Quality** tab, and then verify that the appropriate value is specified in the **DQ Setup Location** field. Then specify the appropriate values in the **DQ Setup Location** field and the **Scheme Repository Type** field. The **DQ Setup Location** field should specify appropriate Quality Knowledge Base (QKB) location value (DQSETUPLOC value). The **Scheme Repository Type** field should match the type (BFD or NOBFD) for the package of jobs that you are importing.

When the package is imported, the job and its transformations are updated as they are during migration. The Apply Lookup Standardization transformation uses the default scheme repository values, and if the values are correct the transformation runs successfully.

## **Updates to Jobs and Transformations during Partial Promotion**

You might have jobs stored in locations that are not part of the migration process. If these jobs can be exported to SAS 9.1.3 Package format, you can use the Import from SAS

Package wizard in SAS Data Integration Studio 4.2 to convert and register these jobs on your metadata server.

If you have jobs that were exported in XML format before SAS Data Integration Studio 3.4., you must either include these jobs in a system migration, or import them in SAS Data Integration Studio 3.4, export them as a SAS Package, then import the SAS Package in SAS Data Integration Studio 4.2.

*Note:* When a job in SAS 9.1.3 Package format is imported with the SAS 9.2 Import from SAS Package wizard, the job and its transformations are updated as they are during migration.

## Changes to the Tree View

After migration, you will see the following changes to the tree view on the left side of the desktop.

The Custom tree is now called the Folders tree. Any user-defined folders in the Custom tree now appear in the Folders tree. If a migrated or imported object was not located in a folder, and it is an object type that requires a folder, it is placed in the Shared Data folder.

My Folder is the private folder of the user who is currently logged on. Metadata in this folder is visible only to the current user and to unrestricted users. When you add a new metadata object, and you want to share it with other users, do not save it to My Folder.

In the Folders tree, you cannot have duplicate objects with the same name in the same folder. Accordingly, after migration or when importing metadata from versions before version 4.2, if there were duplicate objects with the same name in the same folder, a number will be added as a suffix to the duplicate file name (filename(1), filename(2), etc.). Also, metadata objects can no longer have slash characters in their names. If a migrated or imported object had slash characters in its name, the slashes are replaced by underscores.

Unlike the Custom tree, in the Folders tree, you cannot drag and drop objects from one top-level folder to another. Instead, you can right-click the object and select **Move to Folder**.

The Inventory tree now contains folders for more kind of objects. Most of the time, however, SAS Data Integration Studio users work with the same objects as before, such as tables, libraries, and jobs. Some actions, such as importing metadata, can no longer be done in the Inventory tree. You must perform these actions from the Folders tree.

The Process Library tree is now called the Transformations tree. The Transformations tree supports one level of folders only. If transformations in your Process Library were organized in a hierarchy of folders, they now appear at the top level of the Transformations tree.

The Project tree, a special tree that was used under change management, is now called the Checkouts tree.

In previous versions of SAS Data Integration Studio, metadata repositories were the top level objects in the Inventory tree and the Custom tree. Now, metadata repositories are not visible in the Inventory tree, and they are just another folder in the Folders tree. In most cases it is no longer important which metadata repository contains a particular object. If you have the appropriate privilege, you can work with any object in the tree views, regardless of what metadata repository contains the object.

## SAS Workspace Server Requirements for New Jobs

From this release forward, new jobs with code that is generated by SAS Data Integration Studio must be executed on a SAS 9.2 Workspace Server or newer.

### ***Impacts on Change Management***

Administrators who are responsible for setting up change management in SAS Data Integration Studio must do some additional work after migration. For more information, see the SAS Data Integration Studio chapter in the *SAS Intelligence Platform Desktop Application Administration Guide*.

### ***Impacts on SAS Solutions***

Some SAS 9.2 solutions might not be available until after the release of SAS Data Integration Studio 4.2. Accordingly, if your site is using a SAS solution, it is recommended that you not import individual solution objects until that solution is ready for SAS 9.2 and is installed.

Alternatively, you can migrate or import jobs with missing solution transformations and simply not open them until after the SAS 9.2 solution is installed. After the SAS 9.2 solution is installed, you can select a job in a tree view and use the **Upgrade** pop-up menu option to upgrade the job.

### ***Migration Web Site***

For more information about migration, see our Migration Web site: <http://support.sas.com/rnd/migration/utility/utilitynotes>.



## Part 2

---

# General User Tasks

<i>Chapter 2</i>	
<b>Getting Started</b>	21
<i>Chapter 3</i>	
<b>Importing, Exporting, and Copying Metadata</b>	45
<i>Chapter 4</i>	
<b>Working with Tables</b>	63
<i>Chapter 5</i>	
<b>Working with External Files</b>	99
<i>Chapter 6</i>	
<b>Creating Jobs</b>	121
<i>Chapter 7</i>	
<b>Managing Jobs</b>	141
<i>Chapter 8</i>	
<b>Managing the Status of Jobs and Transformations</b>	167
<i>Chapter 9</i>	
<b>Deploying Jobs</b>	183
<i>Chapter 10</i>	
<b>Working with Generated Code</b>	205
<i>Chapter 11</i>	
<b>Working with User-Written Code</b>	213
<i>Chapter 12</i>	
<b>Optimizing Process Flows</b>	231
<i>Chapter 13</i>	
<b>Using Impact Analysis</b>	247
<i>Chapter 14</i>	
<b>Working with Reports</b>	255





## Chapter 2

# Getting Started

---

<b>Setup for SAS Data Integration Studio</b> . . . . .	<b>22</b>
Basic Setup . . . . .	22
<b>Security for SAS Data Integration Studio</b> . . . . .	<b>22</b>
Overview of Security . . . . .	22
Authorization Tab . . . . .	23
<b>Main Tasks for Creating Process Flows</b> . . . . .	<b>23</b>
<b>Starting SAS Data Integration Studio</b> . . . . .	<b>24</b>
Problem . . . . .	24
Solution . . . . .	24
Tasks . . . . .	24
<b>Connecting to a SAS Metadata Server</b> . . . . .	<b>26</b>
Problem . . . . .	26
Solution . . . . .	26
Tasks . . . . .	26
<b>Working with the Folders Tree</b> . . . . .	<b>27</b>
Overview of the Folders Tree . . . . .	27
Add a Folder . . . . .	28
Add Metadata Objects to a Folder . . . . .	29
Copy to Folder . . . . .	29
Drag to Folder . . . . .	29
Move to Folder . . . . .	29
Rename a Folder . . . . .	29
Considerations When You Change a Folder Path . . . . .	30
<b>Selecting a Default SAS Application Server</b> . . . . .	<b>30</b>
Problem . . . . .	30
Solution . . . . .	30
Tasks . . . . .	30
<b>Registering SAS Libraries</b> . . . . .	<b>31</b>
Problem . . . . .	31
Solution . . . . .	31
Tasks . . . . .	31
<b>Working with User-Defined Formats</b> . . . . .	<b>32</b>
Problem . . . . .	32
Solution . . . . .	32
Tasks . . . . .	32
<b>Registering Tables and Cubes</b> . . . . .	<b>33</b>
Problem . . . . .	33

Solution .....	33
Tasks .....	33
<b>Specifying Global Options in SAS Data Integration Studio .....</b>	<b>35</b>
Problem .....	35
Solution .....	35
Tasks .....	35
<b>Working with Change Management .....</b>	<b>36</b>
Problem .....	36
Solution .....	36
Tasks .....	37
<b>Add a Note or Document to a Registered Object .....</b>	<b>41</b>
Problem .....	41
Solution .....	41
Tasks .....	41
<b>View the Content of Notes or Documents .....</b>	<b>43</b>
Problem .....	43
Solution .....	43
Tasks .....	43

---

## Setup for SAS Data Integration Studio

### Basic Setup

SAS Data Integration Studio depends on servers, clients, and other resources in a data integration environment. Administrators install and configure these resources, and SAS Data Integration Studio users are told which resources to use. At a minimum, the following resources must be installed to support SAS Data Integration Studio. For more information about these and other resources, see the installation instructions for your SAS data integration environment.

**Table 2.1** Components Required by SAS Data Integration Studio 4.2 or Later

Component	Description
SAS Metadata Server	SAS 9.2 Metadata Server or later.
SAS Application Server	SAS Application Server with SAS 9.2 server components or later, including a SAS 9.2 Workspace Server.

---

## Security for SAS Data Integration Studio

### Overview of Security

In order to build and execute process flows in SAS Data Integration Studio, you must have privileges such as the following:

- read and write access to the sources and targets in the job, as specified by the operating system and other relevant systems such as database servers
- read and write access to the metadata for sources and targets in the job, as specified on the SAS Metadata Server
- read and write access to folders in the **Folders** tree on the desktop

Typically, SAS Data Integration Studio users use the privileges that are granted to them by a security administrator and do not set security attributes themselves. For example, an administrator can set up the custom folder structure in the **Folders** tree and set permissions on those folders. Most users simply save objects to those folders, without setting any permissions on individual objects.

For details about setting up security, administrators should see the *SAS Intelligence Platform: Security Administration Guide*. The "Permissions on Folders" section describes how to set permissions on folders in the **Folders** tree. Under change management, there are additional security considerations for users and administrators. See [“Working with Change Management”](#) on page 36.

### Authorization Tab

An **Authorization** tab can be displayed in the property windows for tables, libraries, transformations, and many other objects. This tab can be used to view or update the metadata permissions on these objects. In general, users do not set permissions on individual objects, but this capability is available if needed. For more information about using the **Authorization** tab, see the "Working with Permissions" chapter in the *SAS Intelligence Platform: Security Administration Guide*.

Each user can control whether the **Authorization** tab is hidden or displayed in his or her SAS Data Integration Studio session. To toggle the display of this tab, select **Tools** ⇨ **Options** from the menu bar. In the Options window, click the **General** tab, and then select or deselect the **Show advanced property tabs** check box.

---

## Main Tasks for Creating Process Flows

Here are the main tasks for creating process flows in SAS Data Integration Studio:

1. Start SAS Data Integration Studio.
2. Open an existing connection profile or create a new one that connects to the appropriate metadata server.
3. Select a default SAS Application Server.
4. Add metadata for the inputs to a process flow (data sources).
5. Add metadata for the outputs from a process flow (data targets).
6. Create a new job and a process flow that reads the appropriate sources, performs the required transformations, and loads the target data store with the desired information.
7. Run the job.

---

## Starting SAS Data Integration Studio

### Problem

You want to start SAS Data Integration Studio.

### 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 `distudio.ini` file. For more information, see the following tasks:

- [“Start SAS Data Integration Studio” on page 24](#)
- [“Specify Java Options” on page 24](#)
- [“Specify the Plug-in Location” on page 24](#)
- [“Specify the Error Log Location” on page 25](#)
- [“Specify Message Logging” on page 25](#)
- [“Allocate More Memory to SAS Data Integration Studio” on page 25](#)

For more information about command-line arguments for SAS client applications, administrators should see the *SAS Intelligence Platform Desktop Application Administration Guide*.

### Tasks

#### Start SAS Data Integration Studio

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 `distudio.exe` command.

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

```
distudio -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

```
distudio -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:

```
distudio -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 2.2** 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 (**distudio.exe**) is found. There is an **.ini** file with the same name as the executable (**distudio.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 SAS Metadata Server

### Problem

You want to work with tables, jobs, and other objects in SAS Data Integration Studio.

### Solution

Create and open a connection profile, which connects to a SAS Metadata Server. You can then work with tables, jobs, and other objects that have been specified in the metadata, and you can add new metadata as needed.

When you create a connection profile, you can select the **Use Integrated Windows authentication (single sign-on)** option if you know that your environment supports single sign-on. For more information about single sign-on, administrators should see the "Dictionary of Authentication Mechanisms" chapter of the *SAS Intelligence Platform: Security Administration Guide*.

The main tasks for maintaining connection profiles are as follows:

- [“Create a Connection Profile” on page 26](#)
- [“Open a Connection Profile” on page 26](#)
- [“Update a Connection Profile” on page 27](#)
- [“Reconnecting to a Metadata Server” on page 27](#)

### Tasks

#### Create a Connection Profile

Perform the following steps to create a connection profile:

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 the metadata server
2. Start SAS Data Integration Studio. The Connection Profile window displays.
3. Select **Create a new connection profile**. The New Connection Profile wizard displays.
4. Click **Next**, and enter a name for the profile.
5. Click **Next**, and enter a machine address, port, user name, and password that enables you to connect to the appropriate SAS Metadata Server.
6. Click **Finish** to exit the connection profile wizard, connect to the metadata server, and display the server's metadata in SAS Data Integration Studio.

#### Open a Connection Profile

Perform the following steps to open a connection profile that was created earlier:

1. Start SAS Data Integration Studio. The Connection Profile window displays.

2. Select **Open an existing connection profile**.
3. Use the selection arrow to select the profile to be opened, and click **Ok**.

Another way to open an existing connection profile is to start SAS Data Integration Studio, and then select **File** ⇒ **Connection Profile** from the menu bar. The Connection Profile window displays, and you perform the same steps as in the preceding task.

After you open a connection profile, you are connected to the metadata server, and the server's metadata is displayed in SAS Data Integration Studio. If you are working under change management, the name of your project repository is displayed in the **Checkouts** tree on the desktop. If you are not working under change management, you do not see the **Checkouts** tree.

### **Update a Connection Profile**

Perform the following steps to update a connection profile:

1. Start SAS Data Integration Studio. The Connection Profile window displays.
2. Use the selection arrow to select the profile that you want to edit, and then click **Edit**. The Edit Connection Profile wizard displays.
3. Update the profile as needed, and then click **Finish** to exit the connection profile wizard, connect to the metadata server, and display the server's metadata in SAS Data Integration Studio.

### **Reconnecting to a Metadata Server**

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 attempts to reconnect to the metadata server.

If the reconnection is successful, you can continue your work. The user credentials from the previous session is 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.

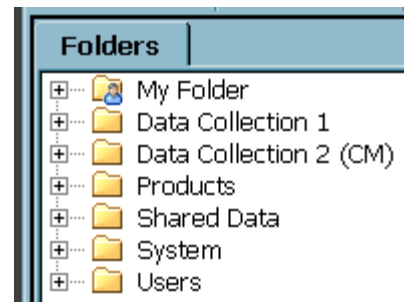
---

## **Working with the Folders Tree**

### **Overview of the Folders Tree**

The Folders tree is one of the tree views in the left panel of the desktop. Like the Inventory tree, the Folders tree displays metadata for objects that are registered on the current metadata server, such as tables and libraries. The Inventory tree, however, organizes metadata by type and does not allow you to add custom folders. The Folders tree enables you to add custom folders.

Some folders in the Folders tree are provided by default, such as **My Folder**, **Products**, **Shared Data**, **System**, and **Users**. Typically, SAS Data Integration Studio users work with metadata in custom folders, such as the **Data Collection 1** folder and **Data Collection 2 (CM)** folder as shown in the following display.

**Display 2.1** Example Folders in the Folders Tree

In general, an administrator sets up the custom folder structure in the Folders tree and sets permissions on those folders. Users simply save metadata to the appropriate folders in that structure. For example, given the folder structure shown in the preceding display, users can save metadata to a sub-folder under **Data Collection 1**. Users who work under change management can save metadata to a sub-folder under **Data Collection 2 (CM)**. Any additions or changes to your custom folder structure should be carefully planned, as described in [“Considerations When You Change a Folder Path”](#) on page 30.

In general, SAS Data Integration Studio users work with the following folders:

- The custom folders, such as the **Data Collection 1** and **Data Collection 2 (CM)** folders in the preceding display, are used to organize metadata that you want to be available to other users. Custom folders are usually added to the root of the tree or to the **Shared Data** folder.
- The **Shared Data** folder is a default folder that can be used to organize metadata that you want to be available to other users. Your site might or might not choose to save metadata to this folder.
- **My Folder** is the private folder of the user who is currently logged on. It is similar to the **My Documents** folder in Microsoft Windows. Metadata in **My Folder** is visible only to the owning user and to unrestricted users, so this folder can be used to store metadata that you are not ready to make available to other users.

When you first begin adding metadata objects in SAS Data Integration Studio, these objects might be added to **My Folder** by default. To make these objects visible to other people who are connected to the same metadata server, you can use the **Move to Folder** option to move the metadata in an appropriate public folder in the Folders tree.

Users who are working under change management should not use **My Folder**. They should use the **Checkouts** tree and the change-managed folder instead. For more information, see [“Working with Change Management”](#) on page 36.

## Add a Folder

Perform the following steps to add a custom folder without selecting a parent folder in the Folders tree.

1. From the desktop select **New** ⇒ **Folder**.
2. Enter a name for the folder. Verify that the folder path in the **Location** field is the path you want. To specify a different path in the Folders tree, click **Browse** and select the desired path.
3. Select **OK** to create the new folder.



Perform the following steps to add a sub-folder to a folder that you select in the Folders tree:

1. Right-click a folder in the Folders tree and select **New** ⇒ **Folder**. An untitled folder is added to the parent folder.
2. Type a new name for the folder.

### Add Metadata Objects to a Folder

When you add a metadata object, it is added to a folder in the Folders tree and in the Inventory tree. You can specify the folder in the Folders where new metadata is added. To save a new metadata object to a specific folder in the Folders tree, right-click that folder, select **New**, and then select the appropriate wizard. Alternatively, if you select **New** from the menu bar, and then select the appropriate wizard, you can use the **Browse** control beside the **Location** field to change the folder path for the new object.

### Copy to Folder

Perform the following steps to create a copy of a metadata object and save that copy to a different folder.

1. Right-click an object in the Folders tree and select **Copy to Folder**.
2. Select a target folder and click **OK**.

### Drag to Folder

You can drag metadata objects from one folder to another folder within a top-level folder. This changes the folder path of the object. See [“Considerations When You Change a Folder Path” on page 30](#).

You cannot drag an object from one top-level folder to another top-level folder. For example, you cannot drag an object from **My Folder** to the **Shared Data** folder. You can use the **Move to Folder** option to perform this task.

### Move to Folder

Use the **Move to Folder** option to move a metadata object from one folder to another folder in the Folders tree. This changes the folder path of the object. See [“Considerations When You Change a Folder Path” on page 30](#).

Perform the following steps to move a metadata object to a different folder.

1. Right-click an object in the Folders tree and select **Move to Folder**.
2. Select a target folder and click **OK**.

### Rename a Folder

You can rename a folder. This changes the folder path of the objects in the folder. See [“Considerations When You Change a Folder Path” on page 30](#).

Perform the following steps to rename a folder.

1. Right-click the folder in the Folders tree and select **Rename**.
2. Enter a new name for the folder.

### **Considerations When You Change a Folder Path**

*Note:* Use caution when renaming folders and when moving objects from one folder to another.

Any additions or changes to your custom folder structure, and any movement of objects from one folder to another, should be carefully planned. Some types of objects are referenced using folder pathnames. Associations to these types of objects can break if you move the object to a different folder. If you break an association based on a folder path, you can restore it by updating the folder path in the affected object.

For example, reports use folder paths to locate information maps. If you move an information map to a different folder, then you might need to edit associated reports to point to the new information map location. Other objects that depend on folder pathnames include information maps and prompts. For more information about managing folder pathnames, see the "Working with SAS Folders" chapter in the *SAS Intelligence Platform: System Administration Guide*.

---

## **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. Alternatively, you can double-click the SAS Application Server pane at the bottom of the desktop, to the left of the user ID panel. (The status bar at the bottom of the desktop displays the current user, SAS Application Server, and SAS Metadata 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 or servers 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.

---

## Registering SAS Libraries

### Problem

You want to register a SAS library so that you can access tables in that library.

### Solution

Use the New Library wizard to register the 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. You cannot use SAS Data Integration Studio to register tables, run jobs that read and write tables, or view data in tables until the libraries that contain these tables have been registered.

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

*Note:* Registering a library does not, in itself, provide access to tables in the library. You must perform a separate operation to register any tables that you want to access in the library. See [“Registering Tables and Cubes” on page 33](#).

### Tasks

#### **Register a SAS Library**

Perform the following steps to register a SAS library:

1. From the SAS Data Integration Studio desktop, select the appropriate folder in the Folders tree, then select **File** ⇒ **New** ⇒ **Library** 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.
2. After you have selected the library type, click **OK**.
3. Enter the rest of the library metadata as prompted by the wizard.

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

---

## Working with User-Defined Formats

### Problem

You want to use the View Data window to display data with user-defined formats, or you want to execute a job that contains a table with user-defined formats.

### Solution

Make user-defined formats available from the SAS Application Server, or make them available for a particular job.

A format is an instruction that SAS uses to write data values. Formats are used to control the written appearance of data values, or, in some cases, to group data values together for analysis. An informat is an instruction that SAS uses to read nonstandard data values, such as dates, currency values, or hexadecimal values.

To make a custom format library available to any application that uses a particular SAS Application Server, administrators should see the "Working With User-Defined Formats" section of the "Connecting to Common Data Sources" chapter in the *SAS Intelligence Platform: Data Administration Guide*.

To make a custom format library available to a specific job, see [“Specify a Format Library in a Preprocess to a Job” on page 32](#).

### Tasks

#### ***Specify a Format Library in a Preprocess to a Job***

SAS Data Integration Studio users can specify the location of the format library in a preprocess to a job. The preprocess would consist of SAS statements such as the following:

```
Options fmtsearch=(myformat library work);  
libname myformat "C:\formats\myformats";
```

The SAS Application Server that executes the job must be able to resolve the path that you specify in the LIBNAME statement for the format library.

The following steps describe one way to specify a format library in a preprocess to a job:

1. From the SAS Data Integration Studio desktop, select the job you want to update, then select **Edit** ⇒ **Properties** from the menu bar. The property window for the job displays.
2. Click the Precode and Postcode tab, and then select the **Precode** check box.
3. In the code panel, enter a FMTSEARCH option and a LIBNAME statement that are similar to the previous example code.
4. To save the precode in metadata, click **OK**. To save the precode to a file, click **Save As**, specify a server and filename for the code, and then click **OK**.

When you execute the job, the preprocess code runs first and the specified format library becomes available when the rest of the job executes.

## Registering Tables and Cubes

### Problem

You want to work with a table or a cube that is not visible in the tree view on the SAS Data Integration Studio desktop.

### Solution

Register the table or cube.

To register an object means to save metadata about that object to a SAS Metadata Server. After you register an object, its metadata is displayed in the tree view. You can then work with that object in SAS Data Integration Studio.

The main tasks for registering tables and cubes are as follows:

- [“Register Tables or Cubes” on page 33](#)
- [“Preserving Foreign Keys in DBMS Tables” on page 35](#)

### Tasks

#### Register Tables or Cubes

Use the methods in the following table to add metadata for tables or cubes in SAS Data Integration Studio.

*Note:* The Register Table wizard and the New Table wizard use a SAS library to access the tables that you want to register. It is simpler if any required libraries are registered before you run these wizards. See [“Registering SAS Libraries” on page 31](#).

**Table 2.3** *Methods for Registering Tables or Cubes*

Objects to be Registered	Method for Specifying Metadata
A set of table metadata in Common Warehouse Metamodel (CWM) format or in a format that is supported by a SAS Metadata Bridge.	Select <b>File</b> ⇒ <b>Import</b> ⇒ <b>Metadata</b> from the menu bar to import the metadata.
A set of table metadata exported from SAS Data Integration Studio as a SAS Package File.	Select an appropriate destination folder in the tree view, and then select <b>File</b> ⇒ <b>Import</b> ⇒ <b>SAS Package</b> from the menu bar to import the metadata.
One or more SAS tables or database management system tables (DBMS) tables that exist in physical storage.	Select <b>File</b> ⇒ <b>Register Tables</b> from the menu bar, select the appropriate format, and then respond to the Register Table wizard. Alternatively, right-click the library that contains the tables to be registered, and then select <b>Register Tables</b> .

Objects to be Registered	Method for Specifying Metadata
A table that is specified in a comma-delimited file or in another external file.	Select <b>File</b> ⇒ <b>New</b> ⇒ <b>External File</b> ⇒ <b>Delimited</b> from the menu bar, select the appropriate external file format, and then respond to the external file wizard.
A new table that is created when a SAS Data Integration Studio job is executed. Or, a new table that reuses column metadata from one or more registered tables.	Select <b>New</b> ⇒ <b>Table</b> from the menu bar, and then respond to the New Table wizard.
One or more tables that are specified in an XML file.	Select <b>File</b> ⇒ <b>Register Tables</b> from the menu bar, select the XML format, and then respond to the Register Tables wizard. For more information, administrators should see the sections about XML in the chapters about common data sources in the <i>SAS Intelligence Platform: Data Administration Guide</i> .
A Microsoft Excel spreadsheet.	Select <b>File</b> ⇒ <b>Register Tables</b> from the menu bar, select the Excel or ODBC format, and then respond to the Register Tables wizard. For more information, administrators should see the sections about ODBC in the chapters about common data sources in the <i>SAS Intelligence Platform: Data Administration Guide</i> .
One or more tables that exist in physical storage and that can be accessed with an Open Database Connectivity (ODBC) driver.	Select <b>File</b> ⇒ <b>Register Tables</b> from the menu bar, select the ODBC format, and then respond to the Register Tables wizard. For more information, administrators should see the sections about ODBC in the chapters about common data sources in the <i>SAS Intelligence Platform: Data Administration Guide</i> .
A table in a format that does not appear in your Register Tables wizard. (Your site might not have licensed all of the formats that are available from SAS.)	Select <b>File</b> ⇒ <b>Register Tables</b> from the menu bar, select the <b>Generic</b> format, and then respond to the Register Table wizard.  The Generic format in the Register Tables wizard uses a Generic Library to access tables. A Generic library enables you to manually specify a SAS engine and the options that are associated with that engine. Because it is general by design, a Generic Library offers few hints as to what options should be specified for a particular engine. Accordingly, a Generic Library might be most useful to experienced SAS users. For details about the options for a particular engine, see the SAS documentation for that engine.

Objects to be Registered	Method for Specifying Metadata
A SAS cube.	Select <b>File</b> ⇒ <b>New</b> ⇒ <b>Cube</b> from the menu bar, and then respond to the New Cube wizard.

### ***Preserving Foreign Keys in DBMS Tables***

Tables in a database management system often have primary keys, unique keys, and foreign keys. When you register a DBMS table with foreign keys, if you want to preserve the foreign keys, select all of the tables that are referenced by the foreign keys at the same time, in a single pass of the wizard. Similarly, when you export or import a DBMS table with foreign keys, select all of the tables that are referenced by the foreign keys at the same time, in a single pass of the wizard.

---

## **Specifying Global Options in SAS Data Integration Studio**

### ***Problem***

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

### ***Solution***

Specify the appropriate option in the start command for SAS Data Integration Studio, or specify an option in the global Options window, as described in the following topics:

- [“Starting SAS Data Integration Studio” on page 24](#)
- [“Use the Global Options Window” on page 35](#)

### ***Tasks***

#### ***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:

- general interface options for SAS Data Integration Studio
- options for the **Diagram** tab of the Job Editor window
- options for the **Code** tab of the Job Editor window
- options for the default SAS Application Server for SAS Data Integration Studio
- options for the View Data window
- options which specify how SAS Data Integration Studio generates code
- data quality options, such as options for the Create Match Codes transformation and the Apply Lookup Standardization transformation

---

## Working with Change Management

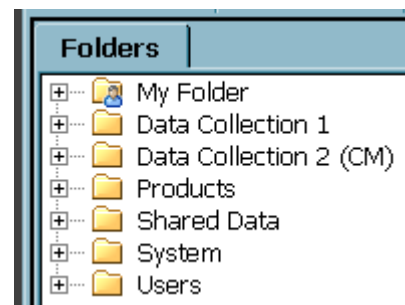
### Problem

A team of SAS Data Integration Studio users wants to work simultaneously with a set of related metadata. They want to avoid overwriting each other's changes.

### Solution

Have an administrator set up a change-managed folder in the Folders tree, such as the **Data Collection 2 (CM)** folder shown in the following display.

**Display 2.2** Data Collection 2 (CM) Folder is Under Change Management

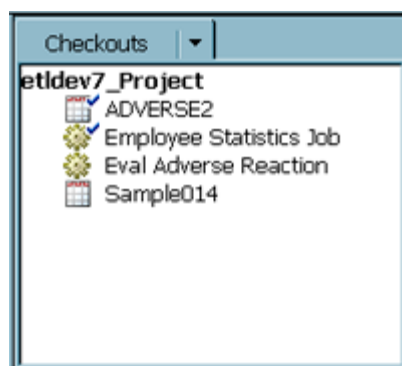


Under change management, most users are restricted from adding or updating the metadata in a change-managed folder in the Folders tree. Authorized users, however, can add new metadata objects and check them in to the change-managed folder. They can also check out metadata objects from the change-managed folder in order to update them. The objects are locked so that no one else can update them as long as the objects are checked out. When the users are ready, they check in the objects to the change-managed folder, and the lock is released.

If you are authorized to work in a change-managed folder, a Checkouts tree is added to your desktop in SAS Data Integration Studio. The Checkouts tree displays metadata in your project repository, which is an individual work area or playpen.

To update a metadata object in the change-managed folder, check out the object. The object is locked in the change-managed folder, and a copy is placed in the Checkouts tree. Metadata that has been checked out for update has a check mark beside it, such as the first two objects in the following display.



**Display 2.3** Sample Checkouts Tree

You can modify the copy in the Checkouts tree. When ready, check in the updated object to the change-managed folder. Any lock on that object is released and any updates are applied.

To add a new metadata object to the change-managed folder, add the object as usual. The metadata is added to the Checkouts tree. New metadata objects that have never been checked in do not have a check mark beside them, such as the last two objects in the preceding display. When ready, check in the new object to the change-managed folder.

*Note:* Users who are working under change management should not use **My Folder** in the Folders tree. They should use the Checkouts tree and the change-managed folder instead.

For, example, when you add a new metadata object, verify that the folder path in the **Location** field for the object goes to the appropriate, change-managed folder. For information about setting up change management, administrators should see the “Administering SAS Data Integration Studio” chapter of the *SAS Intelligence Platform Desktop Application Administration Guide*.

Working with change management involves the following tasks:

- “Create a Connection Profile for a User Under Change Management” on page 37
- “Create a Connection Profile for an Administrator Under Change Management” on page 38
- “Add New Metadata” on page 38
- “Check In Metadata” on page 38
- “Check Out Metadata” on page 39
- “Delete Metadata” on page 39
- “Undo Checkouts” on page 40
- “Clear All Metadata from Your Project” on page 40
- “Clear All Metadata from a Project That You Do Not Own” on page 40

See also “Usage Notes for Change Management” on page 40.

## Tasks

### **Create a Connection Profile for a User Under Change Management**

Perform the following steps to create a connection profile that enables you to work with metadata in a change-managed folder:

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 that enable you to work in a change-managed folder
  - the name of the project that you specify in your connection profile
2. Start SAS Data Integration Studio. The Connection Profile window displays.
3. Select **Create a new connection profile**. The New Connection Profile wizard displays.
4. Click **Next**, and enter a name for the profile.
5. Click **Next**, and enter a machine address, port, user name, and password that 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 Project Selection page displays.
7. Select the appropriate project. Then select the **Connect to a project** check box.
8. Click **Finish** to exit the connection profile wizard, connect to the metadata server, and display the server's metadata in SAS Data Integration Studio. The name of your project repository is displayed in the **Checkouts** tree on the desktop.

### **Create a Connection Profile for an Administrator Under Change Management**

The standard set of privileges that enable you to work in a change-managed folder do not enable you to perform administrative tasks such as the following:

- deploy a job for scheduling
- deploy a job as a stored process
- create a Web service from a stored process
- clear a project repository that you do not own

In order to perform tasks such as these, you must use a connection profile that has appropriate privileges in the change-managed folder. Ask an administrator for a logon ID and password that has the privileges you need for these tasks. Then create and use the connection profile as usual.

### **Add New Metadata**

Perform the following steps to add a new metadata object to a change-managed folder:

1. If you have not done so already, open a connection profile that enables you to work with the metadata in a change-managed folder.
2. Add the metadata as usual. Verify that the folder path in the **Location** field for the object goes to the appropriate, change-managed folder. To specify a different path in the Folders tree, click **Browse** and select the desired path. The new object appears in the **Checkouts** tree on the desktop. The new object is not displayed in other trees until it is checked in for the first time.
3. When you are finished working with the new metadata, you can check it in to the change-managed folder.

### **Check In Metadata**

Perform the following steps to check in metadata to a change-managed folder:

1. To check in selected objects, select one or more objects in the **Checkouts** tree, right-click them, and select **Check In**. The Check In Wizard displays.

Alternatively, to check in all metadata in your project, right-click the name of the project in the Checkouts tree, and select **Check In All**. The Check In Wizard displays.

2. In the Check In Wizard, enter a title and an optional description for the changes that you are about to check in. The text entered here becomes part of the history for all objects that you are checking in. If you do not enter meaningful comments, the history is less useful. When you are finished describing your changes, click **Next**. The Select Objects to Check In page displays.

You can use the Select Objects to Check In page to identify any checked-out objects that depend on an object that you selected for check-in. For example, suppose that you had checked out a job and also a table that was in the process flow for that job. If you selected the job for check-in, the Select Objects to Check In page would indicate that a table in that job was also checked out. In that case, you might want to check it in along with the job.

3. To skip the Select Objects to Check In page, click **Next** to display the Finish window.

Otherwise, select an object in the Select Objects to Check In page. Any checked-out objects that depend on the object that you just selected are displayed on the **Dependencies** tab. Use the **Dependencies** and other tabs on this page to determine whether you want to check in a dependent object along with the parent object. When finished, click **Next** to display the Finish window.

4. Review the metadata and click **Finish** to check in the metadata.

After check in, any new or updated metadata that was in your **Checkouts** tree is moved to the change-managed folder.

### **Check Out Metadata**

Perform the following steps to check out metadata from a change-managed folder:

1. If you have not done so already, open a connection profile that enables you to work with the metadata in a change-managed folder.
2. In the change-managed folder, right-click the metadata that you want to check out and select **Check Out**. Alternatively, you can left-click the metadata that you want to check out, then go the menu bar, and select **Check Outs** ⇒ **Check Out**. The metadata is checked out and displays in your Checkouts tree.

After you are finished working with the metadata, you can check it in to the change-managed folder.

### **Delete Metadata**

You can use the **Delete** option to permanently remove selected metadata objects from the metadata server. Metadata objects that have never been checked in are simply deleted from the Checkouts tree. Metadata objects that are checked out are deleted from the metadata server.

*Note:* Metadata objects that are deleted cannot be recovered except by restoring the metadata repository from backup.

Perform the following steps to permanently remove selected metadata objects.

1. If the metadata objects that you want to delete are not checked out, check them out.
2. In the Checkouts tree, select one or more objects that you want to permanently remove.

3. Right-click the object or objects and select **Delete**.
4. Click **Yes** when prompted to verify the delete operation.

### **Undo Checkouts**

You can use the **Undo Checkout** option to discard any changes to selected metadata objects that have been checked out. The objects are removed from the Checkouts tree, and the original objects are unlocked in the change-managed folder. Any changes made to the metadata since it was checked out are lost. Perform the following steps to undo checkouts:

1. In the Checkouts tree, select one or more checked-out objects whose changes should be discarded.
2. Right-click the object or objects and select **Undo Checkout**.
3. Click **Yes** when prompted to verify the undo checkout operation.

### **Clear All Metadata from Your Project**

You can use the **Clear** option to delete all new objects and unlock all checked-out objects in your Checkouts tree. You can use this option any time that you want to discard all new and updated metadata in your **Checkouts** tree. You can also use this option when a metadata object fails to check in due to technical problems. When you clear a project, all changes that have not been checked in are lost. Perform the following steps to use this option:

Right-click the Checkouts tree and select **Clear**. Alternatively, you can select the name of your project in the **Checkouts** tree, then select **Checkouts** ⇒ **Clear** from the menu bar.

### **Clear All Metadata from a Project That You Do Not Own**

Problems can occur that require an administrator to clear all metadata from a user's project repository, which is the metadata repository that populates the Checkouts tree. For example, suppose 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. As another example, suppose an administrator accidentally deletes a user's project repository that contains checked-out objects. These objects would remain locked and unavailable for update until they were unlocked.

If problems such as these occur, an administrator can perform the following steps to clear all metadata from one or more project repositories:

1. Start SAS Data Integration Studio. Select a connection profile for an unrestricted user, as described in [“Create a Connection Profile for an Administrator Under Change Management”](#) on page 38.
2. On the SAS Data Integration Studio desktop, select **Checkouts** ⇒ **Clear** from the menu bar. The Clear Project Repository window displays. Unrestricted users see all project repositories on the current metadata server.
3. If the project repository that you want to clear been deleted, select **Search for deleted project repository information**. Any deleted project repositories on the current metadata server are listed.
4. In the Clear Project Repository window, select one or more project repositories to be cleared. Then, click **OK**. In the selected projects, all new objects are deleted, and all checked-out objects are unlocked. All changes that have not been checked in are lost.

### **Usage Notes for Change Management**

Under change management, you can neither add new cubes nor check out existing cubes for update.

Under change management, there is limited support for the following kinds of objects: Stored Processes, Information Maps, Web Services, Deployed Jobs, Deployed Flows, Mining Results, Reports, and Prompts. You can add these objects and check them in once. You can import these objects and check them in once. However, some actions might not be supported for these objects.

---

## Add a Note or Document to a Registered Object

### Problem

The metadata for libraries, tables, and other registered objects includes a **Description** field. This field is limited to 200 characters, but some objects might need a longer description.

### Solution

You can type text into the **Quick Note** field on the **Notes** tab on the properties window for the object. Alternatively, you can create a note or document and associate it with the metadata for the object that you want to describe.

Notes are generally short and contain only minimal formatting. A document is usually longer, and it might have been authored using a word-processing program or a desktop-publishing application. Documents can contain more elaborate formatting, graphics, and so on.

Use the following methods to add notes or documents to the metadata for a library, table, or another object:

- [“Add a Quick Note to a Metadata Object” on page 41](#)
- [“Create a Note and Attach It to a Metadata Object” on page 41](#)
- [“Create a Document and Attach It to a Metadata Object” on page 42](#)
- [“Attach One or More Registered Notes or Documents to a Metadata Object” on page 42](#)
- [“Associate a Quick Note, a Note, or a Document with a Column” on page 43](#)

### Tasks

#### **Add a Quick Note to a Metadata Object**

Perform the following steps to add a quick note to a metadata object:

1. In a SAS application, display the properties window for the object that you want to describe.
2. Click the **Notes** tab.
3. Type the desired text into the **Quick Notes** field.
4. Click **OK** to save your changes.

#### **Create a Note and Attach It to a Metadata Object**

Perform the following steps to create a note and associate it with a metadata object:

1. In a SAS application, display the properties window for the resource that you want to describe.
2. Click the **Notes** tab.
3. In the **Notes** area of the tab, click **New**. The New Notes window displays.
4. In the **Name** field, enter a name for the metadata to identify the note.
5. (Optional) In the **Description** field, enter a longer description for the metadata to identify the note.
6. In the **Location** field, accept the default folder or click the **Browse** button to select the folder in the **Folders** tree. The metadata for the note is stored in the selected folder.
7. In the **Text** field, enter a note that describes the current object.
8. Click **OK** to save your changes and associate the note with the current object.

### **Create a Document and Attach It to a Metadata Object**

Perform the following steps to create a document and associate it with a metadata object:

1. Use third-party software to create a document that describes one or more registered objects. Remember the path to the document.
2. In a SAS application, display the properties window for an object that you described in Step 1.
3. Click the **Notes** tab.
4. In the **Documents** area of the tab, click **New**. The New Documents window displays.
5. In the **Name** field, enter a name for the metadata that identifies the document.
6. (Optional) In the **Description** field, enter a longer description for the metadata that identifies the document.
7. In the **Location** field, accept the default folder or click the **Browse** button to select the folder in the **Folders** tree. The metadata for the document is stored in the selected folder.
8. Click the right corner of the **Path** field to display the file selection button and click that button. A file selection window displays for the default SAS Application Server or a SAS Application Server that you select.
9. Use the file selection window to select the document that you created in Step 1.
10. Click **OK** to save your changes and associate the selected document with the current object.

### **Attach One or More Registered Notes or Documents to a Metadata Object**

Perform the following steps to associate one or more registered notes or documents with a metadata object:

1. In a SAS application, display the properties window for the metadata object.
2. Click the **Notes** tab.
3. In the **Notes** area or the **Documents** area of the tab, click **Attach**. The Select Notes window or the Select Documents window displays.
4. In the window, use the **Folders** tree to display the desired notes or documents. Select one or more notes or documents, and then click the right arrow to move them into the **Selected** column.
5. Click **OK** to link the selected notes or documents to the current metadata object.

**Associate a Quick Note, a Note, or a Document with a Column**

Perform the following steps to associate a quick note, a note, or a document with the metadata for a table column:

1. In a SAS application, display the properties window for a table with a column that you want to describe with a quick note, a note, or a document.
2. Click the **Columns** tab.
3. Right-click the column that you want to describe, and then select **Properties**. The column properties window displays.
4. Attach a quick note, a note, or a document, as described in the previous tasks.

---

## View the Content of Notes or Documents

**Problem**

You want to view the quick notes that have been added to a registered object, or you want to view the content of notes or documents that are registered on the current metadata server.

**Solution**

Use one of the following methods:

- [“View Quick Notes, Notes, or Documents Associated with a Registered Object” on page 43](#)
- [“View Notes in the SAS Data Integration Studio Tree View” on page 43](#)
- [“View Documents in the SAS Data Integration Studio Tree View” on page 44](#)

**Tasks****View Quick Notes, Notes, or Documents Associated with a Registered Object**

Display the properties window for the object and click the **Notes** tab. Quick notes are displayed in the **Quick Notes** field.

For a note, select the note from the **Notes Assigned** list, and the text of the note displays in the **Note text** area.

For a document, make note of the specified path for the document in which you are interested. You need third-party software to open the actual document.

**View Notes in the SAS Data Integration Studio Tree View**

SAS Data Integration Studio supports the following method for displaying the contents of a registered note:

1. In the tree view, right-click the note and select **Properties**.
2. Click the **Details** tab to read the contents of the note.

### ***View Documents in the SAS Data Integration Studio Tree View***

SAS Data Integration Studio supports the following method for displaying the contents of a registered document:

1. In the tree view, right-click the document and select **Open** to read the contents of a document in HTML format and some other formats.
2. If the document is not displayed, right-click the document and select **Properties**.
3. Click the **Details** tab. Note the specified path for the document. You need third-party software to open the actual document.



## Chapter 3

# Importing, Exporting, and Copying Metadata

---

<b>Metadata Import and Export in SAS Data Integration Studio</b>	<b>46</b>
<b>Working with SAS Package Metadata</b>	<b>46</b>
About Importing and Exporting SAS Package Metadata	46
Objects That Can Be Imported and Exported in SAS Package Format	47
Importing Earlier Versions of SAS Package Metadata	47
<b>Preparing to Import or Export SAS Package Metadata</b>	<b>47</b>
<b>Exporting SAS Package Metadata</b>	<b>48</b>
Problem	48
Solution	48
Tasks	48
<b>Importing SAS Package Metadata</b>	<b>49</b>
Problem	49
Solution	49
Tasks	50
<b>Copying and Pasting Metadata Objects</b>	<b>51</b>
Problem	51
Solution	51
Tasks	51
<b>Working with SAS Metadata Bridges</b>	<b>51</b>
About SAS Metadata Bridges	51
Objects That Can be Imported or Exported with a SAS Metadata Bridge	52
<b>Usage Notes for Importing or Exporting with a SAS Metadata Bridge</b>	<b>52</b>
<b>Preparing to Import or Export with a SAS Metadata Bridge</b>	<b>53</b>
<b>Importing New Metadata with a SAS Metadata Bridge</b>	<b>53</b>
Problem	53
Solution	53
Tasks	53
<b>Importing Updated Metadata with a SAS Metadata Bridge</b>	<b>55</b>
Problem	55
Solution	55
Tasks	56
<b>Exporting Metadata with a SAS Metadata Bridge</b>	<b>60</b>
Problem	60
Solution	60
Tasks	61

---

## Metadata Import and Export in SAS Data Integration Studio

SAS Data Integration Studio enables you to import and export metadata for individual objects or sets of related objects. You can work with two kinds of metadata:

- SAS metadata in SAS Package format
- relational metadata (metadata for libraries, tables, columns, indexes, and keys) in formats that can be accessed with a SAS Metadata Bridge

By importing and exporting SAS Package metadata, you can move the metadata for SAS Data Integration Studio jobs and related objects between SAS Metadata Servers. For example, you can create a job in a test environment, export it as a SAS Package, and import it into another instance of SAS Data Integration Studio in a production environment.

By importing and exporting relational metadata in external formats, you can reuse metadata from third-party applications, and you can reuse SAS metadata in those applications as well. For example, you can use third-party data modeling software to specify a star schema for a set of tables. The model can be exported in Common Warehouse Metamodel (CWM) format. You can then use a SAS Metadata Bridge to import that model into SAS Data Integration Studio.

This chapter focuses on the wizards that are used to import and export individual objects or sets of related objects in SAS Data Integration Studio. For a more comprehensive view of metadata management, administrators should see the metadata management chapters in the *SAS Intelligence Platform: System Administration Guide*. See also the technical paper, “Metadata Promotion in SAS 9.2” at: [http://support.sas.com/resources/papers/tnote/tnote\\_migration.html](http://support.sas.com/resources/papers/tnote/tnote_migration.html).

---

## Working with SAS Package Metadata

### ***About Importing and Exporting SAS Package Metadata***

The SAS Intelligence Platform provides tools that enable you to promote individual metadata objects or groups of objects from one metadata server to another, or from one location to another on the same metadata server. You can also promote the physical files that are associated with the metadata.

The promotion tools include:

- the Export to SAS Package wizard and the Import from SAS Package wizard, which are available in SAS Data Integration Studio, SAS Management Console, and SAS OLAP Cube Studio.
- the batch import tool and the batch export tool, which enable you to perform promotions on a scheduled or repeatable basis. These tools provide most of the same capabilities as the SAS Package import and export wizards. For information about the batch import tool and the batch export tool, see the “Using the Promotion Tools” chapter in the *SAS Intelligence Platform: System Administration Guide*.

The SAS Package import and export wizards enable you to reuse the metadata for tables, jobs, and other objects. For example, you can develop a job in a test environment, export

it, and then import the job into a production environment. These wizards enable you to perform the following tasks:

- export the metadata for one or more selected objects in a tree view.
- export the metadata for all objects in one or more selected folders in the Folders 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 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.

### **Objects That Can Be Imported and Exported in SAS Package Format**

You can import and export SAS Package metadata for any object type that is included in the SAS Data Integration Studio Inventory tree. For a description of these objects, see [“Inventory Tree” on page 404](#).

### **Importing Earlier Versions of SAS Package Metadata**

If you are migrating from an earlier version to SAS Data Integration Studio 4.2, administrators need to migrate your metadata servers. However, you might have jobs stored in locations that are not part of the migration process. If these jobs can be exported to SAS 9.1.3 Package format, you can use the Import from SAS Package wizard in SAS Data Integration Studio 4.2 to convert and register these jobs on your metadata server.

If you have jobs that were exported in XML format before SAS Data Integration Studio 3.4., you must either include these jobs in a system migration, or import them in SAS Data Integration Studio 3.4, export them as a SAS Package, and then import the SAS Package in SAS Data Integration Studio 4.2.

*Note:* When a job in SAS 9.1.3 Package format is imported with the SAS 9.2 Import from SAS Package wizard, the job and its transformations are updated as they are during migration.

For details about the impacts of conversion, see [“Updates to Jobs and Transformations During Migration” on page 14](#) and [“User Action Required for Migrated Jobs with Data Quality Transformations” on page 15](#).

---

## **Preparing to Import or Export SAS Package Metadata**

The SAS Package import and export wizards are easy to use, especially when you are working with small packages of metadata on the same metadata server. However, it can sometimes be difficult to map servers, libraries, and other attributes when an object is imported from a different metadata server. Accordingly, administrators should carefully plan the import or export of large amounts of metadata, or the import of metadata from one metadata server to another. For more information, administrators should see the "Using the Promotion Tools" chapter in the *SAS Intelligence Platform: System Administration Guide*.

---

## Exporting SAS Package 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 the same metadata server or to a different metadata server. The source and target server can be located on the same host machine or on different host machines. It is assumed that you have prepared for this task as described in [“Preparing to Import or Export SAS Package Metadata”](#) on page 47.

Perform the following tasks:

- [“Document the Metadata That Will Be Exported \(optional\)”](#) on page 48
- [“Export Selected Metadata”](#) on page 48

### Tasks

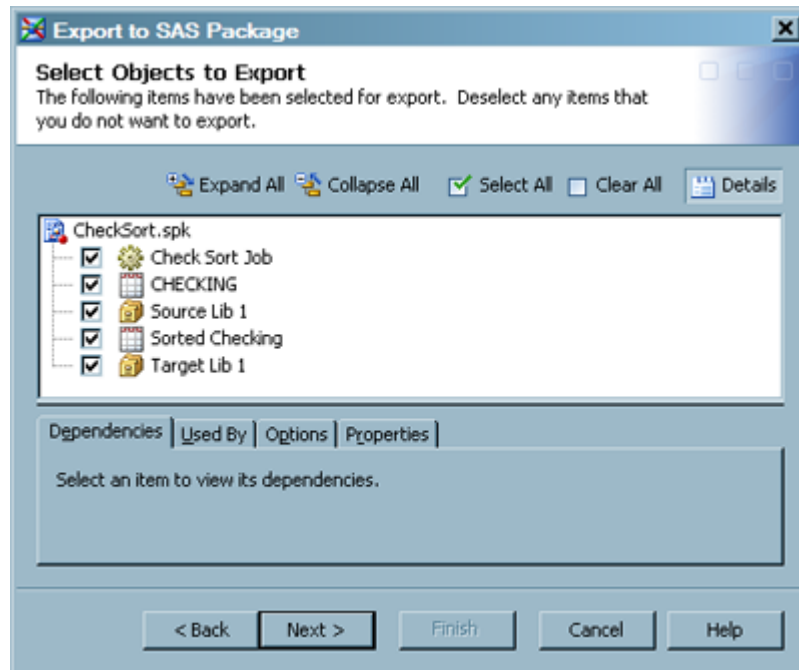
#### ***Document the Metadata That Will Be Exported (optional)***

Metadata export and import tasks are 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 import and export wizards for SAS Packages.

#### ***Export Selected Metadata***

Perform the following steps to export metadata using a SAS package:

1. In the tree view, right-click the objects to be exported and select **Export** ⇒ **SAS Package** from the pop-up menu. The Export SAS Package Wizard displays. Alternatively, you can left-click the objects to be exported and select **File** ⇒ **Export** ⇒ **SAS Package** from the menu bar.
2. In the first page of the wizard, specify a path and name for the export package or accept the default. If you want to include dependent objects when you create the package, you can click the **Include dependent objects when retrieving initial collection of objects** check box. For example, you can export a job named Check Sort and name the package CheckSort.spk. The full pathname for the sample job is `C:\export\CheckSort.spk`. When you are finished, click **Next** to access the Select Objects to Export page.
3. 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. You can click **Details** in the toolbar to see tabs at the bottom of the page. These tabs enable you to review dependencies, information, options, and properties for a selected object. The Select Objects to Export page is shown in the following display.

**Display 3.1** Select Objects to Export Page

Click **Next** to access the Summary page.

4. Review the metadata to be exported. Then, click **Next**. 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.
5. If desired, click **View Log** to view a log of the export operation. When you are finished, click **Finish**.

---

## Importing SAS Package Metadata

### Problem

You want to import metadata into SAS Data Integration Studio that was exported in SAS Package format.

### Solution

Use the Import to SAS Package wizard to import the SAS package file that contains the metadata. The package can be saved to the same metadata server or to a different metadata server. The source and target server can be located on the same host machine or on different host machines. It is assumed that you have prepared for this task described in [“Preparing to Import or Export SAS Package Metadata” on page 47](#).

## Tasks

### ***Identify the Metadata That Should Be Imported (optional)***

It is 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 from SAS Package wizard prompts 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 to import metadata using a SAS package:

1. In the Folders tree, right-click the folder into which metadata should be imported and select **Import** from the pop-up menu. The Import wizard displays. Alternatively, you can left-click a folder and select **File** ⇒ **Import** ⇒ **SAS Package** from the menu bar.
2. 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 are not registered on the target metadata server). When finished, click **Next** to access the Select Objects to Import page.
3. 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.
4. 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 you want to import content, if content was exported with the object. You can also click **Properties** to review its properties. When finished, click **Next** to access the About Metadata Connections page.
5. Review any metadata associations to be restored. For example, if you are importing a table, you are prompted to specify a library for that table. Click **Next** to access the SAS Application Servers page and begin restoring the required associations.
6. Review any application server associations. Then, click **Next** to access the Directory Paths page.
7. Review any directory paths. Then, click **Next** to access the Summary page.
8. Review the metadata to be imported. Then click **Next** to access the Importing Object page. 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.
9. If desired, click **View Log** to view a log of the import operation. When finished, click **Finish**.

---

## Copying and Pasting Metadata Objects

### **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, and then modify the copy as desired. As an alternative to **Paste**, you can use **Paste Special**, 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**

**Paste** enables you to create a copy that is almost identical to the original that you copied. To paste an object, right-click a target folder in the Folders tree object and select **Paste** from the pop-up menu.

#### **Paste Special**

**Paste Special** enables you to select which attributes are copied and to change some attributes in the pasted object. Right-click a target folder in the Folders tree, then select **Paste Special** from the pop-up menu.

---

## Working with SAS Metadata Bridges

### **About SAS Metadata Bridges**

SAS Data Integration Studio can import and export relational metadata in any format that is supported by a SAS Metadata Bridge. By importing and exporting relational metadata in external formats, you can reuse metadata from third-party applications, and you can reuse SAS metadata in those applications as well. For example, you can use third-party data modeling software to specify a star schema for a set of tables. The model can be exported in Common Warehouse Metamodel (CWM) format. You can then use a SAS Metadata Bridge to import that model into SAS Data Integration Studio.

The Export Metadata Wizard enables you to export relational metadata from SAS Data Integration Studio to a file, in any format that is supported by a SAS Metadata Bridge. The Import Metadata Wizard enables you to perform the following tasks:

- Import relational metadata in a file, in any format that can be accessed with a SAS Metadata Bridge.

- 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.

### ***Objects That Can be Imported or Exported with a SAS Metadata Bridge***

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

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

---

## **Usage Notes for Importing or Exporting with a SAS Metadata Bridge**

- 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 perform 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.
- If you are working under change management, empty your Checkouts tree of any metadata before importing more metadata with the Import Metadata Wizard. This makes it easier to manage the imported metadata from a particular session. If you want to save any metadata in the Checkouts tree, check in that metadata. If you want to discard any remaining metadata in the Checkouts tree, you can select **Check Outs** ⇒ **Clear Repository** from the menu bar.
- The Import Metadata 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.
- When imported metadata is compared to existing metadata, and you are working under change management, imported metadata is compared to the checked-in metadata. Accordingly, any metadata in the Checkouts tree that has not been checked in is not 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 Checkouts tree and then select **Comparison Recompare** from the toolbar in the Differences window.

- 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 with a SAS Metadata Bridge

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 New Metadata with a SAS Metadata Bridge

### **Problem**

You want to import metadata for one or more tables that have never been registered on the current metadata server. The metadata is in a format that is accessible with a SAS Metadata Bridge.

### **Solution**

You can use the Import Metadata Wizard and select the **Import as new metadata** option on the Import Selection page. This option specifies that metadata in the selected file is imported without comparing it to existing metadata.

*Note:* The **Import as new metadata** option eliminates some steps, but it can result in duplicate metadata, if any of the metadata that you are importing is for an object that has already been registered on the current metadata server.

Under change management, the imported metadata appears in your Checkouts tree, where you can review it before checking it in. Without change management, all metadata in the selected file is registered to the target metadata server.

### **Tasks**

#### **Import As New Metadata**

The following preparation makes it easier to import metadata as new:

- Identify the folder in the Folders tree that contains the imported metadata. You can create a new folder, if you need to do so.
- Identify the path to the file that contains the metadata to be imported.
- Identify the library in the target environment that contains the imported metadata. You can register a new library, if you need to do so.

Follow these steps to import metadata that is in a format that can be accessed by a SAS Metadata Bridge. The Common Warehouse Metamodel (CWM) format is one example.

Perform the following steps to import metadata for one or more tables that have never been registered on the current metadata server:

1. Right-click the folder in the Folders tree that stores the imported metadata. Then, select **Import** ⇒ **Metadata** to access the Select an import format page of the Metadata Import Wizard. This page lists the formats that are licensed for your site.
2. Verify that the folder specified in the **Folders** field on the File Location page is the folder that you designated as the storage location for the imported metadata. If the folder is incorrect, click **Browse** to select a different folder.
3. Specify a path to the file that contains the metadata to be imported in the **File name** field. The path must be accessible to the default SAS Application server or to a server you select with the **Advanced** button on this page. Click **Next** to access the Meta Integration Options page.
4. Review the information on the Meta Integration Options page. Typically, you accept the default values.

*Note:* The Meta Integration Options page enables you to specify how the wizard imports various kinds of metadata in the source file. To see a description of each option, select the option in the **Name** field, and a description of that option appears in the pane at the bottom of the page. Typically, you can accept the defaults on this page. The following display shows the Meta Integration Options page for the sample job.

**Display 3.2** Meta Integration Options

Name	Value
Target Tool	Auto Detect
Auto Correct	True
Top Package	Logical View
Import UUIDs	True
Data model Tables design level	Physical
Dimensional model reverse engineering	Disabled

\*Required option

Specify which tool was used to generate the model you want to import. This parameter allows the bridge to fine tune the its behavior to match the way the source tool saved the model.

'Auto Detect' - The bridge will auto-detect which tool generated the file.

'YMAC CWM' - Specifies that the file conforms to the CMC CWM standard DTD.

< Back   Next >   Finish   Cancel   Help

Click **Next** to access the Import Selection page.

5. The Import Selection page enables you to select whether the metadata is imported as new or compared to existing metadata in the target environment. Because the sample job is a new metadata import, select **Import as new metadata**. Then, click **Next** to access the Metadata Location page.

6. The Metadata Location page enables you to specify the library in the target environment that should contain the imported metadata. If necessary, you can click the ellipsis button in the **Library** field to select the library. The content in the **DBMS** and **Schema** fields is based on the library that you select. Click **Next** to access the Finish page.
7. Review the metadata. Click **Finish** to import the metadata. When prompted to view the import log, respond as needed. After you skip or view the log, the Import Metadata wizard will close. Verify that the metadata was imported to the appropriate library and folder.

If you are not working under change management, all tables that are specified in the imported metadata are registered to the target metadata repository. Verify that the table metadata was imported into the correct folder and library.

Also, be aware that if you are working under change management, the imported tables might not appear in the Checkouts tree until you refresh the tree. Right-click the Checkouts tree and select **Refresh**.

---

## Importing Updated Metadata with a SAS Metadata Bridge

### **Problem**

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

### **Solution**

You can use the Import Metadata Wizard and select the **Compare import metadata to repository** option on the Import Selection page. This option specifies that metadata in the selected file is imported and compared to existing metadata. Differences in tables, columns, indexes, and keys are detected.

Under change management, imported metadata is compared to the checked in metadata that is associated with the library 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. Differences are stored in a comparison result library. You can view the changes in the Differences window.

Perform the following tasks:

- [“Import the Metadata to be Compared” on page 56](#)
- [“Compare the Imported Metadata to the Existing Metadata” on page 58](#)
- [“Applying Changes to Tables with Foreign Keys” on page 59](#)
- [“Restoring Metadata for Foreign Keys” on page 60](#)
- [“Deleting an Invalid Change Analysis Result” on page 60](#)

## Tasks

### **Import the Metadata to be Compared**

The following preparation makes it easier to import the metadata that you need to compare to existing metadata:

- Identify the folder in the Folders tree that contains the existing metadata that are updated with the imported metadata.
- Identify the path to the file that contains the metadata to be imported.
- Identify the library that contains the differences between the imported metadata and existing metadata (the comparison result library). Register a new library, if necessary.
- Identify the library in the target environment that contains the imported metadata. Register a new library, if necessary. (This library is generally created when the library metadata is first imported.)

Perform the following steps to compare imported metadata to existing metadata:

1. Right-click the folder in the Folders tree that stores the imported metadata. Then, select **Import ⇒ Metadata** to access the Select an import format page of the Metadata Import Wizard. This page lists the formats that are licensed for your site.

*Note:* If you select the wrong folder, the imported metadata is not compared to the appropriate existing metadata. Some or all of the imported metadata might then show up incorrectly as new in the Differences window.

2. From the Metadata Import Wizard, select the format of the file that you want to import. For example, a sample job could use the commonly used OMG CWM (Common Warehouse Metamodel) format. Click **Next** to access the File Location page.
3. Specify a path to the file that contains the metadata to be imported in the **File name** field. The path must be accessible to the default SAS Application server or to a server that you select with the **Advanced** button on this page. Click **Next** to access the Meta Integration Options page.
4. Review the information on the Meta Integration Options page. Typically, you accept the default values.

*Note:* The Meta Integration Options page enables you to specify how the wizard imports various kinds of metadata in the source file. To see a description of each option, select the option in the **Name** field, and a description of that option appears in the pane at the bottom of the page. Typically, you can accept the defaults on this page. The following display shows the Meta Integration Options page for the sample job.

**Display 3.3** Meta Integration Options

Name	Value
Target Tool	Auto Detect
Auto Correct	True
Top Package	Logical View
Import UUIDs	True
Data model Tables design level	Physical
Dimensional model reverse engineering	Disabled

\*Required option

Specify which tool was used to generate the model you want to import. This parameter allows the bridge to fine tune the its behavior to match the way the source tool saved the model.

\*Auto Detect - The bridge will auto-detect which tool generated the file.

\*OMAC CDM - Specifies that the file conforms to the OMAC CDM standard DTDs.

< Back   Next >   Finish   Cancel   Help

Click **Next** to access the Import Selection page.

5. The Import Selection page enables you to select whether the metadata is imported as new or compared to existing metadata in the target environment. Because the sample job compares the imported metadata to existing metadata, select **Compare import metadata to repository**.
 

*Note:* If the wizard detects that the metadata to be imported is similar to existing metadata in the folder that you selected when you began the import, it selects **Compare import metadata to repository** by default. If this option is not selected, select it now. The **Comparison results library** field becomes active.
6. Use the drop-down menu to select a comparison result library in the **Comparison results library** field. You can change the default options for the comparison by clicking **Advanced** to display the Advanced Comparison Options window. Click **Next** to access the Metadata Location page.
7. The Metadata Location page enables you to specify the library in the target environment that should contain the imported metadata. You should select the same library that contains the existing metadata that is compared to the imported metadata. If necessary, you can click the ellipsis button in the **Library** field to select the library. Note that the content in the **DBMS** and **Schema** fields is based on the library that you select. Click **Next** to access the Finish page.
8. Review the metadata. Click **Finish** to import the metadata. When prompted to view the import log, respond as needed. After you skip or view the log, the Import Metadata wizard will close. Verify that the metadata was imported to the appropriate library and folder.
9. 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 Checkouts tree, right-click the **Project repository** icon and select **Check In Repository**.

If you are not working under change management, all tables that are specified in the imported metadata are registered to the target metadata repository. Verify that the table metadata was imported into the correct folder and library.

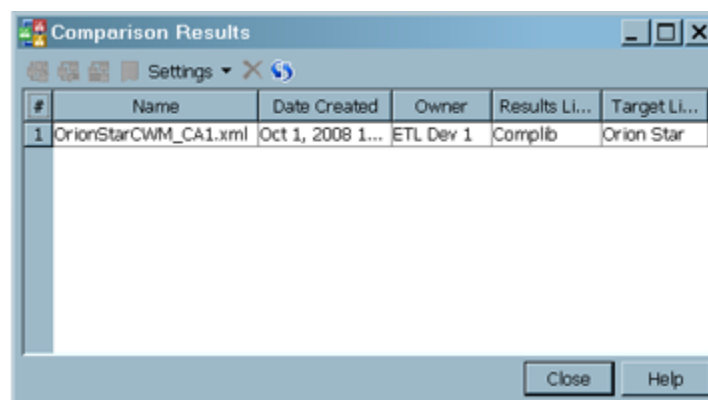
Also, be aware that if you are working under change management, the imported tables might not appear in the Checkouts tree until you refresh the tree. Right-click the Checkouts tree and select **Refresh**.

### **Compare the Imported Metadata to the Existing Metadata**

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

1. Select **Tools** ⇒ **Comparison Results** from the menu bar on the desktop to access the Comparison Results window. The following display shows the Comparison Results window for a sample job.

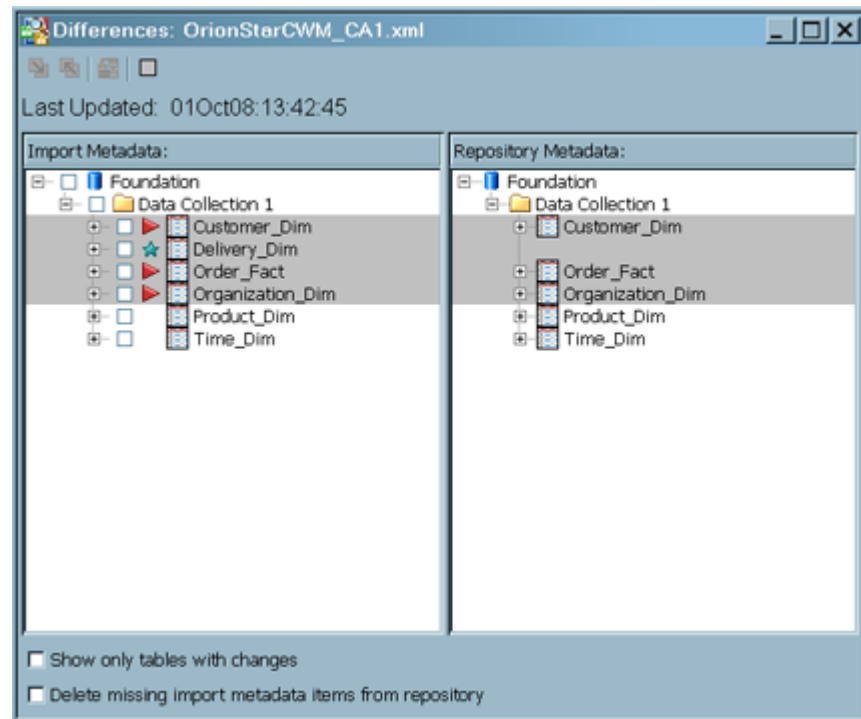
**Display 3.4** Comparison Results Window



The Comparison Results window enables you to select the results of a compare import metadata to repository operation. There is one record for each successful comparison operation. Select the desired comparison record. Then, click the **View differences found** icon in the toolbar to access the Differences window.

*Note:* The comparison results object is named after the imported file, and it has an XML extension.

2. Expand the folders in the Differences window to determine whether any metadata has changed. A sample Differences window is shown in the following display.

**Display 3.5** Differences Window

Continue to expand folders and view the metadata until you are satisfied that you understand the differences between existing metadata and the imported metadata. To perform impact analysis or reverse impact analysis on an item, select the check box by that item, then click the **Impact Analysis** or **Reverse Impact Analysis** icons on the toolbar on the Differences window. (For a detailed description of all options and controls in the Differences window, press F1.) In this example, the triangle icons in the next display indicate that the imported metadata contains updates to three tables. The star icon indicates that the imported metadata contains one new table.

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 the checked-in metadata displays. Without change management, the Repository Metadata pane displays any matching metadata in the default repository.

3. To apply a change, select the check box next to it in the Differences window. Then click the **Applies the checked changes** icon in the toolbar. A dialog box displays, prompting you to verify the change.
4. Click **OK** to accept the changes. The selected changes are applied. When finished, close the Differences window and the Comparison Results window.

### ***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
- apply selective changes, making sure to select all tables that are related by primary keys or foreign keys

Otherwise, the key relationships are not 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 compare the imported metadata to the metadata in the repository. In the Comparison Results window, select the icon for the appropriate comparison result. Then, click **Redo the comparison** in the toolbar. 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 are not 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 invalid comparison result metadata.

If you are working under change management, perform the following steps to delete an invalid change analysis result:

1. Check in the invalid comparison result metadata. From the Checkouts tree, right-click the **Project** repository icon and select **Check In Repository**. This makes 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 invalid comparison result metadata.
3. Have the administrator delete the invalid comparison result from the Comparison Results tree. This deletes both the metadata and the data sets for a comparison result.

---

## **Exporting Metadata with a SAS Metadata Bridge**

### **Problem**

You want to export metadata from SAS Data Integration Studio in a format that is supported by a SAS Metadata Bridge. For example, you can export metadata for use in a third-party data modeling application. Some SAS solutions rely on this method.

*Note:* This method does not export the metadata to a SAS Package. For information about SAS Packages, see [“Working with SAS Package Metadata” on page 46](#).

### **Solution**

Use the Metadata Export wizard to export the metadata. Later, you can import the metadata in a third-party application or in SAS Data Integration Studio. It is assumed that you have



prepared for this task as described in “Preparing to Import or Export SAS Package Metadata” on page 47.

Perform the following tasks:

- “Document the Metadata That Will Be Exported (optional)” on page 61
- “Export Selected Metadata” on page 61

## Tasks

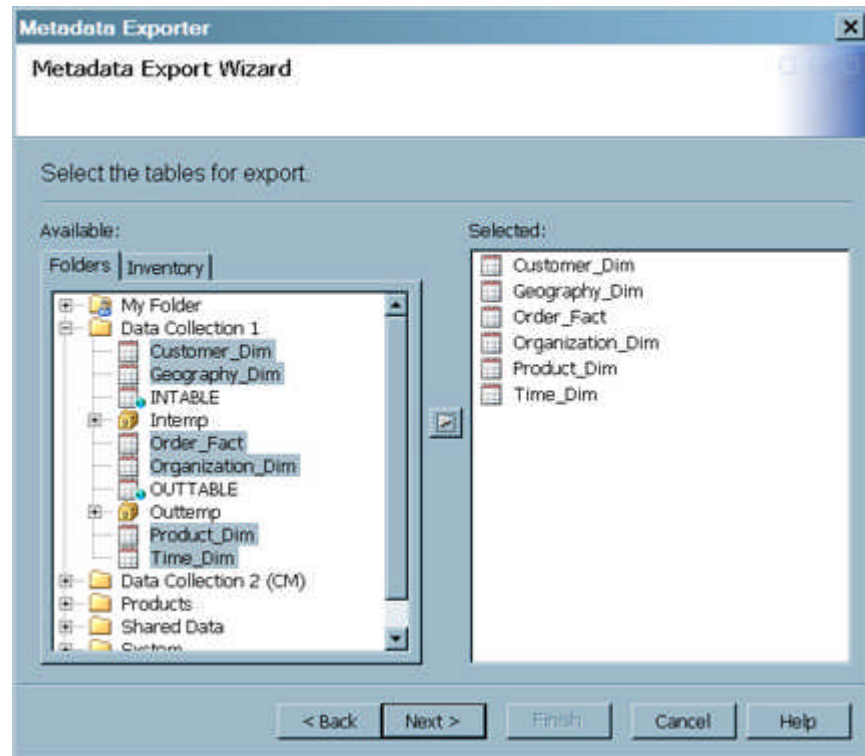
### ***Document the Metadata That Will Be Exported (optional)***

Metadata export and import tasks are 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 import and export wizards.

### ***Export Selected Metadata***

Perform the following steps to export metadata from SAS Data Integration Studio in a format that is supported by a SAS Metadata Bridge.

1. Select **File** ⇒ **Export** ⇒ **Metadata** in the menu bar of the desktop to access the Select an export format page of the Metadata Export Wizard.
2. From the Metadata Import Wizard, select the format of the file that you want to import. For example, a sample job could use the commonly used OMG CWM (Common Warehouse Metamodel) format. Click **Next** to access the Select the tables for export page.
3. Navigate through the folder structure on Select the tables for export page until you locate the tables that you need to export. Then, select the tables in the **Available** field and move them to the **Selected** field. The following display shows the completed Select the tables for export page for a sample job.

**Display 3.6** Select the Tables for Export Page

Click **Next** to access the Specify the file to export the metadata to page.

4. Specify a path and name for the export file. The path and name specify the destination for the exported metadata. Click **Next** to access the Specify Meta Integration Options page.
5. Review the information on the Meta Integration Options page. Typically, you accept the default values.

*Note:* The Meta Integration Options page enables you to specify how the wizard imports various kinds of metadata in the source file. To see a description of each option, select the option in the **Name** field, and a description of that option appears in the pane at the bottom of the page. Typically, you can accept the defaults on this page.

6. Click **Next** to access the Finish page.
7. Review the format and path information for the metadata export. Then, click **Finish** to complete the export process.

## Chapter 4

# Working with Tables

---

<b>About Tables</b> .....	<b>64</b>
<b>Registering Existing Tables with the Register Tables Wizard</b> .....	<b>65</b>
Problem .....	65
Solution .....	65
Tasks .....	65
<b>Registering New Tables with the New Table Wizard</b> .....	<b>66</b>
Problem .....	66
Solution .....	66
Tasks .....	66
<b>Viewing or Updating Table Metadata</b> .....	<b>68</b>
Problem .....	68
Solution .....	68
<b>Using a Physical Table to Update Table Metadata</b> .....	<b>69</b>
Problem .....	69
Solution .....	69
Tasks .....	69
<b>Specifying Options for Tables</b> .....	<b>70</b>
Problem .....	70
Solution .....	70
Tasks .....	70
<b>Supporting Case and Special Characters in Table and Column Names</b> .....	<b>72</b>
Overview .....	72
About Case and Special Characters in SAS Names .....	73
About Case and Special Characters in DBMS Names .....	74
Set Default Name Options for New Tables .....	76
Set Name Options in the Register Tables Wizard .....	76
Set Name Options for Registered Tables .....	76
<b>Maintaining Column Metadata</b> .....	<b>77</b>
Problem .....	77
Solution .....	77
Tasks .....	77
<b>Maintaining Keys</b> .....	<b>82</b>
Problem .....	82
Solution .....	82
Tasks .....	83
<b>Maintaining Indexes</b> .....	<b>87</b>
Problem .....	87

Solution .....	87
Tasks .....	87
<b>Browsing Table Data .....</b>	<b>89</b>
Problem .....	89
Solution .....	89
Tasks .....	89
<b>Editing SAS Table Data .....</b>	<b>92</b>
Problem .....	92
Solution .....	92
Tasks .....	92
<b>Using the View Data Window to Create a SAS Table .....</b>	<b>95</b>
Problem .....	95
Solution .....	95
Tasks .....	95
<b>Specifying Browse and Edit Options for Tables and External Files .....</b>	<b>96</b>
Problem .....	96
Solution .....	96
Tasks .....	96

---

## About Tables

Tables are the inputs and outputs of most SAS Data Integration Studio jobs. The tables can be SAS tables or tables created by the database management systems that are supported by SAS Access software.

The most common tasks for data tables are listed in the following table.

**Table 4.1** Common Table Tasks

Task	Action
Register a table (add metadata about the table's physical location, columns, and other attributes).	For more information, see <a href="#">“Registering Existing Tables with the Register Tables Wizard”</a> on page 65 and <a href="#">“Registering New Tables with the New Table Wizard”</a> on page 66.
Specify a registered table as a source or a target in a job.	Select the table in a tree. Then, drag it to the Job Editor window for the job and connect it to an appropriate input or output port. For more information, see <a href="#">“Creating a Process Flow for a Job”</a> on page 124.
View the data or metadata for a registered table.	For more information, see <a href="#">“Browsing Table Data”</a> on page 89 and <a href="#">“Viewing or Updating Table Metadata”</a> on page 68.

---

## Registering Existing Tables with the Register Tables Wizard

### **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 Register Tables wizard to register the tables. Later, you can drag and drop this metadata into a process flow. When the process flow is executed, SAS Data Integration Studio uses the metadata for the table to access the physical instance of that table.

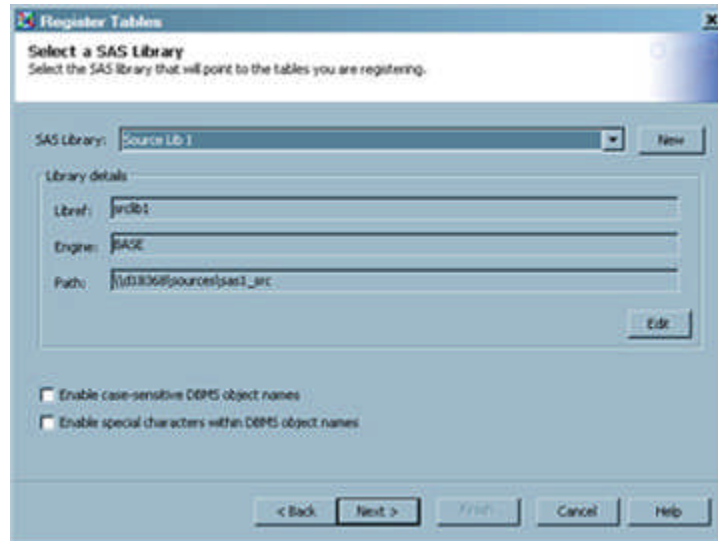
The first page of the wizard prompts you to select a library that contains the tables to be registered. (Typically, this library has been registered ahead of time.) SAS Data Integration Studio must be able to access this library. This library can point to a location that is remote to the current default workspace server, provided that the library is on a system that has an available SAS/CONNECT definition so that remote access can be implemented to that server. This allows for registering tables on systems that do not have a workspace server component.

### **Tasks**

#### **Register a Table with the Register Tables Wizard**

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

1. Display the Register Tables wizard in one of the following ways:
  - Right-click a folder in the **Folders** tree where metadata for the table should be saved, and then select **Register Tables** from the pop-up menu.
  - Select **File** ⇒ **Register Tables**.
  - Right-click a library and select **Register Tables**. Note that the procedure for registering a table in the previous two options begins with a page that asks you to "Select the type of tables that you want to import information about". This page is skipped when you register a table through a library.
2. When the Register Tables wizard opens, only those data formats that are licensed for your site are available for use. Select the data format of the tables that you want 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 Library window.
4. Select the library that contains the tables that you want to register, and review the settings that are displayed in the **Library Details** section of the window. Sample settings for a SAS table are shown in the following display.

**Display 4.1** Sample Library Settings

You can handle case-sensitive and special characters in tables and column names by selecting the respective check box.

5. Click **Next** to access the Define Tables and Select Folder Location page. Select one or more tables to register. Select a folder location, if needed.
6. Click **Next** to access the "The following metadata will be created" page. Review the metadata that is created. When you are satisfied that the metadata is correct, click **Finish** to save the data and close the wizard.

---

## Registering New Tables with the New Table Wizard

### Problem

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

### Solution

Use The New 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 uses the metadata for the target table to create a physical instance of that table.

The physical storage page of the wizard prompts you to select a library that contains the table to be registered. (Typically, this library has been registered ahead of time.)

### Tasks

#### **Register a New Table with the New Table Wizard**

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

1. Display the New Tables wizard in one of the following ways:
  - Right-click the folder in the **Folders** tree where metadata for the new table should be saved. Then select **New** ⇒ **Table**.
  - Select **File** ⇒ **New** ⇒ **Table**.
  - Select **New** ⇒ **Table** on the SAS Data Integration Studio toolbar.

The New Table wizard opens.

2. 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 specify a name for the physical table in a later window in this wizard.
3. Verify that the folder in the Location field is the folder where the metadata for the table should be stored. If not, click **Browse** to select the correct folder.
4. Click **Next** to access the Table Storage Information page. 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 page 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 by 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 4.2** Sample Table Storage Settings

The screenshot shows the 'New Table' wizard window with the 'Table Storage Information' tab selected. The title bar says 'New Table'. Below the title bar, it says 'Table Storage Information' and 'Specify physical storage information about your new target table.' The form contains the following fields and controls:

- DBMS:** A dropdown menu with 'SAS' selected.
- Library:** A dropdown menu with 'Target LB 1' selected. To its right are 'New' and 'Edit' buttons.
- Name:** A text field containing 'Total\_Sales\_By\_Employee'.
- Schema:** An empty text field.
- Two checkboxes:
  - ☐ Enable case-sensitive DBMS object names
  - ☐ Enable special characters within DBMS object names
- A 'Table Options' button at the bottom right.
- Navigation buttons at the bottom: '< Back', 'Next >', 'First', 'Cancel', and 'Help'.

You can handle case-sensitive and special characters in tables and column names by selecting the respective check box.

5. Click **Next** to access the Select Columns page. Use the Select Columns page to import column metadata from existing tables that are registered for use in SAS Data Integration Studio.
6. 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.
7. Click **Next** to access the Change Columns/Indexes page. Use this window to accept or modify any column metadata that you selected in the Select Columns page. You can add new columns or modify existing columns in various ways. (For details, click the **Help** button for the window.)
8. 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.
9. Click **Next** to access the page labeled as The following metadata is created. Review the created metadata. When you are satisfied that the metadata is correct, click **Finish** to save the data and close the 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
- Parameters
- Physical Storage
- Notes
- Extended Attributes
- Authorization

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 Job Editor window. Then, click **Properties** to access its properties window.

Note that updates that you make to the metadata about the table affect all other users of that table's metadata. However, the physical table is not actually updated until you run a job process that actually updates that table. In the case of existing physical tables, in order



to make the physical table match the metadata, it is necessary to drop and recreate 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 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.

---

## Using a Physical Table to Update Table Metadata

### **Problem**

You want to ensure that the metadata for a table matches the physical table.

### **Solution**

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

For existing tables, the update table metadata feature adds new columns, keys and indexes, removes deleted columns, keys, and indexes, and records changes to all of the column, key, and index 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.

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

### **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, right-click one of the tables and select **Update Metadata** in the pop-up 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 that are 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 that are available on the properties window for each table.

### Tasks

#### Set Global Options for Tables

**Table 4.2** Global Table Options

Option Name	Description
Enable case-sensitive DBMS object names	Specifies whether SAS Data Integration Studio generates code when registering and using the table in jobs 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 SAS Data Integration Studio generates code when registering and using the table in jobs 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 <b>Enable case-sensitive DBMS object names</b> 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. For more information about DBMS object names, see [“Supporting Case and Special Characters in Table and Column Names” on page 72](#).

#### 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 4.3** 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. The DBMSs that are valid in the current context are listed.
Library	Specifies a library that you can use to access the table. To select a library, click the selection arrow. To create a new library, click <b>New</b> , which opens the New Library wizard. To edit the properties of the existing library, click <b>Edit</b> , 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 <b>DBMS</b> 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 name might contain a variable. For example, suppose you have the variable &myvar. You can name the table &myvar. You can also imbed the variable in the name (for example, September_&myvar). This usage ensures that parameters that are 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.
Enable case-sensitive DBMS object names	Specifies whether SAS Data Integration Studio generates 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	Specifies whether SAS Data Integration Studio software generates 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 <b>Enable case-sensitive DBMS object names</b> check box is also automatically selected. This option overrides the global option with the same name.
Create as view	Select this option to specify the current table as a view. Deselect this option to specify the current table as a physical table. In the context of a SAS Data Integration Studio job, if <b>Create as view</b> is selected for an output table, and the transformation that creates the table can create views, then the table is created as a view. Some transformations do not support views and might ignore the setting.
Table Options	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 72 for more information about DBMS object names.

You can set additional SAS 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 4.4** Local Table Options in the Table Options Window

Option name	Description
Compress	<p>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. In particular, you 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:</p> <ul style="list-style-type: none"> <li>• <b>NO</b> (default): The SAS data set is not compressed.</li> <li>• <b>YES</b>: The SAS data set is compressed using Run Length Encoding (RLE). Use this method for character data.</li> <li>• <b>BINARY</b>: The SAS data set is compressed by using Ross Data Compression (RDC). Use this method for medium to large (several hundred bytes or larger) blocks of binary data (numeric variables).</li> </ul>
Encrypt	<p>Specifies whether a SAS data set is encrypted (YES) or not encrypted (NO). You cannot encrypt SAS data views because they contain no data.</p>
Additional options	<p>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 <b>read=readpw write=writepw</b>. When using SAS 9.2 as your workspace or batch server, passwords can be encoded using SAS 9.2 encoding.</p>

**Table 4.5** DBMS Table Options in the Table Options Window

Option Name	Description
Table Options	<p>Specifies the appropriate SAS/ACCESS table options for the specific DBMS. Separate each option with a space. The field is restricted to a maximum of 200 characters.</p>

---

## Supporting Case and Special Characters in Table and Column Names

### Overview

The following topics describe how to support case and special characters in table and column names:

- [“About Case and Special Characters in SAS Names” on page 73](#)
- [“About Case and Special Characters in DBMS Names” on page 74](#)
- [“Set Default Name Options for New Tables” on page 76](#)
- [“Set Name Options in the Register Tables Wizard” on page 76](#)
- [“Set Name Options for Registered Tables” on page 76](#)

## 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. In filerefs you can use only 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 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 table options, as described in [“Set Name Options for Registered Tables” on page 76](#) or [“Set Default Name Options for New Tables” on page 76](#). Double-byte character set (DBCS) column names are supported in this way, for example.

The DBMS name options apply to all SAS and DBMS table types, with a few exceptions for SAS tables. The following special rules apply to SAS tables:

- Special characters are not supported in SAS table names.

- Leading blanks are not supported for SAS column names and are removed if you used them.
- Neither the External File wizards 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
- in the metadata for the table itself

For more information, see [“Enable Name Options for a New Database Library” on page 74](#) or [“Enable Name Options for an Existing Database Library” on page 75](#). 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. You should check your database to see whether it supports case-sensitive names and names with special characters.

### Verify Case and Special Character Handling Options for Database Libraries

Perform the following steps to verify that the appropriate DBMS name options have been set for all database libraries where you want to support case and special character handling for tables:

1. Select the library that you want to verify. To easily locate libraries, you can expand the **Libraries** folder in the Inventory tree.
2. 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, verify that both the **Preserve DBMS table names** option is set to **YES** and the **Preserve column names as in the DBMS** option have been set correctly.
3. If these options are not set correctly, update the metadata for the library, as described in [“Enable Name Options for an Existing Database Library” on page 75](#).

### Enable Name Options for a New Database Library

The following task describes how to specify name options for a new relational database library such as Oracle, Sybase, and Teradata. These name options ensure that table and column names are supported as they are 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*. Perform the following steps to specify name options:

1. From the desktop, select **New** ⇒ **Library**. The New Library Wizard opens.
2. In the first window of the New Library wizard, select the appropriate kind of database library and click **Next**.
3. Enter a name for the library and click **Next**.
4. Enter a SAS LIBNAME for the library, and then click **Advanced Options**. The Advanced Options window displays.
5. In the Advanced Options window, click the **Output** tab. In the **Preserve column names as in the DBMS** field, select **Yes**.
6. Click **OK** and enter the rest of the metadata as prompted by the wizard.

### ***Enable Name Options for an Existing Database Library***

Perform the following steps to update the existing metadata for a database library in order 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 desktop, select the Inventory tree.
2. In the Inventory tree, open the **Jobs** folder.
3. Right-click a job that contains DBMS tables and select **Open** from the pop-up menu. The job opens in the Job Editor 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 [“Set Name Options for Registered Tables” on page 76](#).

### Set Default Name Options for New Tables

You can set default name options for all table metadata that is entered with the Register Tables wizard or the New Tables 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 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 the Register Tables wizard or the New Table wizard in SAS Data Integration Studio:

1. Start SAS Data Integration Studio.
2. Open the connection profile that specifies the metadata server where the tables are registered.
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 the Register Tables wizard and the New 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 the Register Tables wizard and the New Table wizard support special characters in table and column names by default.
7. Click **OK** to save any changes.

### Set Name Options in the Register Tables Wizard

The second page in the Register Tables wizard for a DBMS table enables you to select the library that contains the 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**.

### Set Name Options for Registered Tables

Perform the following steps to enable name options for tables that have been registered on a metadata server. 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.

---

## Maintaining Column Metadata

### Problem

You want to add or modify column metadata for registered tables, temporary work tables, and external files.

### Solution

You can use the **Columns** tab 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” on page 77](#)
- [“Modify Metadata for a Column” on page 78](#)
- [“Add and Maintain Notes and Documents for a Column” on page 79](#)
- [“Perform Additional Operations on Column Metadata” on page 80](#)

### Tasks

#### **Add Metadata for a Column**

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

1. Open the properties window for the table or external file, and click the **Columns** tab. The metadata for the current columns, if any, appears in an ordered list.
2. To add metadata for a new column to the end of the current list of columns, click the **New column** icon in the toolbar at the top of the **Columns** tab. Alternatively, you can right-click in a blank area of the **Columns** tab and select **New column** from the pop-up menu.

To insert metadata for a new column after the metadata for a current column, right-click the metadata for the current column, and then select **New column** from the pop-up menu.

After you perform these actions, a row of default metadata that describes the new column displays. The name of the column, **Untitledn**, is selected and ready for editing. The other attributes of the column have the following default values:

- Description: Blank
- Type: Character
- Length: 8
- Informat: (None)
- Format: (None)
- Is Nullable: Yes
- Summary Role: (None)
- Sort Order: (None)

3. Change the name of the column to give it a meaningful name.
4. Change the values of other attributes for the column as desired. For more information, see “[Modify Metadata for a Column](#)” on page 78.
5. Click **OK** to save the new column metadata.

*Note:* You can add columns only when the columns table in the **Columns** tab is sorted on the # column.

### **Modify Metadata for a Column**

To modify the metadata for a column in the current table, open the properties window for the table or external file, and click the **Columns** tab. Select the attribute that you want to change, make the change, and then click **OK**. The following table explains how to change each type of attribute.

**Table 4.6** Column Metadata Modifications

Attribute	Description	Instructions
Name	The SASColumnName of the column. This matches the physical name.	Perform the following steps to enter a name: <ol style="list-style-type: none"> <li>1. Double-click the current name to make it editable.</li> <li>2. Enter a new name of 32 characters or fewer.</li> <li>3. Press the ENTER key.</li> </ol>
Description	This can be the label of the column, and shows up as the label in the generated code.	Perform the following steps to enter a description: <ol style="list-style-type: none"> <li>1. Double-click in the Description field.</li> <li>2. Edit the description, using 200 characters or fewer.</li> <li>3. Press the ENTER key.</li> </ol>
Type	The type can be either numeric or character.	Perform the following steps to enter the data type: <ol style="list-style-type: none"> <li>1. Double-click the current value to display the drop-down list arrow.</li> <li>2. Click the arrow to make a list of valid choices appear.</li> <li>3. Select a value from the list.</li> </ol>
Length	This is the length of the column.	Perform the following steps to enter the column length: <ol style="list-style-type: none"> <li>1. Double-click the current length.</li> <li>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.</li> <li>3. Press the ENTER key.</li> </ol>

Attribute	Description	Instructions
Informat	This specifies a pattern or set of instructions that SAS uses to determine how data values in an input file should be interpreted.	Perform the following steps to enter an informat: <ol style="list-style-type: none"> <li>1. Double-click the current value to display the drop-down list arrow.</li> <li>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 ENTER.</li> </ol>
Format	This specifies a pattern or set of instructions that SAS uses to determine how to display information.	Perform the same steps as for informat.
Is Nullable	This is used to determine whether the integrity constraint IsNullable is set for a specific column. This determines whether a column might have a value of null.	Perform the same steps as for type.
Summary Role	This is used for information purposes only.	Perform the same steps as for type.
Sort Order	This is used for information purposes only.	Perform the same steps as for type.

You can also edit a value by tabbing to it and pressing the F2 key 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 [“Perform Additional Operations on Column Metadata” on page 80](#).

### **Add and Maintain Notes and Documents for a Column**

The **Columns** tab 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 that are 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. Perform the following steps to display this window:

1. Right-click the column that you want to work with and click **Properties** in the pop-up menu. Then, click **Notes** to access the **Notes** tab for the selected column.
2. Perform one or more of the following tasks in the **Notes** group box:
  - Enter the text in the **Quick Note** field. Quick notes are private to this column, while the other type of notes are shared notes.
  - 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.
4. Click **OK** to save the contents of the note.

### **Perform Additional Operations on Column Metadata**

The following table describes some additional operations you can perform on metadata in the **Columns** tab.

**Table 4.7** Additional Operations on Column Metadata

Task	Action
Delete Metadata for a Column	<p>Perform the following steps to delete the metadata for a column in the current table:</p> <ol style="list-style-type: none"> <li>1. Select a column.</li> <li>2. Click <b>Delete</b>.</li> </ol> <p>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.</p> <p>Changes to column metadata in SAS Data Integration Studio do not appear in the physical table automatically. You must select the <b>Replace in the Load Style</b> field and the <b>Entire table in the Replace</b> field on the <b>Load Technique</b> tab of the Table Loader transformation that loads the current table.</p> <p>Column level impact analysis can help you gather information about deleting metadata for a column. To perform impact analysis, right-click on a table and select <b>Analyze</b>. Note that you can also obtain information about reverse impact analysis on another tab in the same window.</p>

Task	Action
Import Metadata for a Column	<p>Perform the following steps to import column metadata that has been added to the metadata server that is specified in your current connection profile:</p> <ol style="list-style-type: none"> <li>1. Click Import columns to access the Import Columns window.</li> <li>2. Locate the table with columns that you want to import. Select one or more columns from the Available field in the Import Columns window.</li> <li>3. Select the right arrow to move the selected columns into the Selected field.</li> <li>4. Reorder the columns in the Selected Columns field by selecting columns and clicking the Moves selected items up or Moves selected items down arrows.</li> <li>5. Click OK to import the columns into the table.</li> </ol> <p>Be aware of the following implications if you add or import metadata for a column:</p> <ul style="list-style-type: none"> <li>• You might need to propagate that column metadata through the job or jobs that include the current table.</li> <li>• Changes to column metadata in SAS Data Integration Studio do not appear in the physical table automatically. You must select the <b>Replace in the Load Style</b> field and the <b>Entire table in the Replace</b> field in the <b>Load Technique</b> tab of the Table Loader transformation that loads the current table.</li> </ul>
Maintain Indexes	<p>Indexes are registered automatically when using Register tables to register metadata about existing tables. In SAS 9.2, indexes are imported correctly when import/export is used. Update table metadata also updates indexes. See <a href="#">“Maintaining Indexes” on page 87</a>.</p>
Maintain Keys	<p>Primary, foreign, and unique keys are registered automatically when using Register tables to register metadata about existing tables. In SAS 9.2, keys are imported correctly when import/export is used. Update table metadata also updates them, although currently it doesn't handle foreign key updates.</p> <p>It is important when working with foreign keys to include ALL of the tables that are related in a single registration. Otherwise, foreign key relationships cannot be maintained. See <a href="#">“Maintaining Keys” on page 82</a>.</p>
Propagate Column Metadata from One Table to Other Tables in a Job	<p>See <a href="#">“Managing the Scope of Column Changes in Jobs” on page 158</a>.</p>

Task	Action
Reorder Columns and Rows	You can rearrange the columns in a table (without sorting them) by dragging a column to a new location. You can reorder rows by (1) using the arrow buttons at the top of the window, or (2) dragging a column to a new location by dragging the column-number cell.
Restore the Order of Columns	Click the column number heading to restore all of the rows to their original order.
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 clicking the Name heading. To sort the columns in descending order by name, you can click the same heading a second time.
View or update extended attributes for columns	From the <b>Columns</b> tab, select the desired column, then click the Properties icon in the toolbar. In the properties window, click the <b>Extended Attributes</b> tab. Use this tab to view or update extended attributes.

---

## Maintaining Keys

### Problem

You want to view, add, or update keys for a table.

### Solution

You can use the **Keys** tab in the properties window for a table to maintain keys. See [“Understanding Keys in SAS Data Integration Studio” on page 83](#). Then perform the following tasks as needed:

- [“View Keys” on page 83](#)
- [“Add a Primary Key or a Unique Key” on page 85](#)
- [“Add a Foreign Key” on page 85](#)
- [“Update the Columns in a Key” on page 86](#)
- [“Delete or Rename a Key” on page 87](#)

## Tasks

### Understanding Keys in SAS Data Integration Studio

SAS Data Integration Studio enables you to manage the following types of keys:

- primary key: a column or combination of columns that uniquely identifies a row in a table. A table can have only one primary 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.
- foreign key: a column or combination of columns in one table that references a corresponding key in another table. A foreign key must have the same data type as the key that it references.

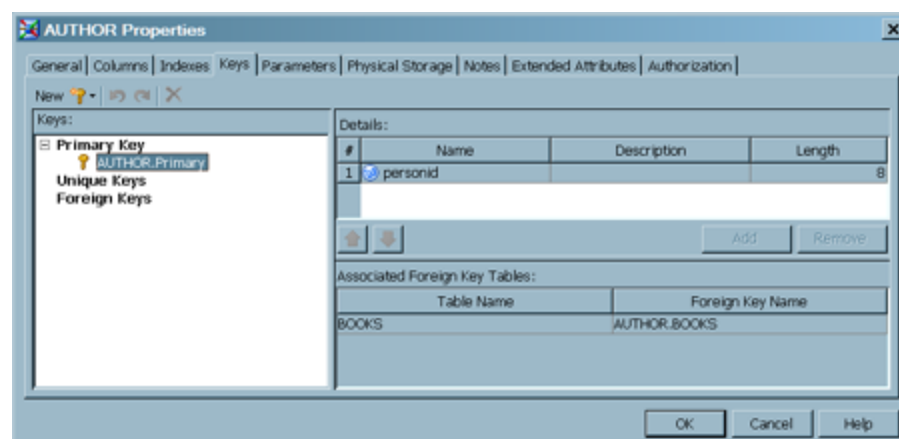
Primary keys and unique keys are often used in table joins. A foreign key is used to create and enforce a link between the data in two tables. A link is created between two tables such that the column or columns that hold a primary key value or a unique key value in one table are referenced by a column or columns in a second table. The column or set of columns in the second table is a foreign key.

*Note:* Some databases, such as Oracle and DB2, support foreign key references to columns in the same table.

### View Keys

To display information about keys that have been specified for a table, access the **Keys** tab on the properties window for the table. On the **Keys** tab, the Keys pane on the left lists all of the keys that are associated with the current table. Click a key in the list to see information about it in the panes on the right: the Details pane and the Associated Foreign Key Tables pane. The following display shows the **Keys** tab for a table named AUTHOR. A primary key named AUTHOR.Primary is selected on the left. Information about this key is shown on the right.

**Display 4.3** Keys Tab with a Primary Key



The default name for a primary key is **currentTableName.Primary**, where **currentTableName** is the name of the current table, and *Primary* is a literal string. For example, the default name for the primary key in the AUTHORS table is AUTHOR.Primary.

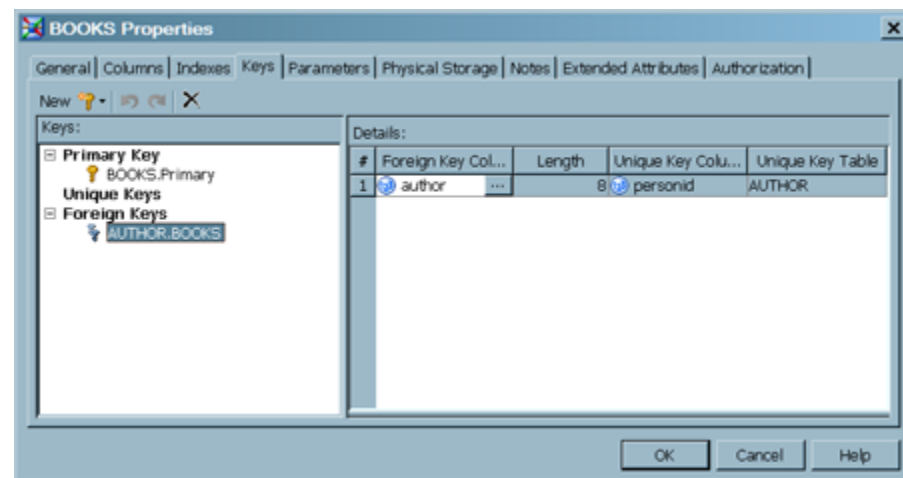
The default name for a unique key is **currentTableName.UniqueKeyN**, where **currentTableName** is the name of the current table, *UniqueKey* is a literal string, and **N** is an iteration number added to the end.

When a primary key or a unique key is selected in the Keys pane, then the columns that are specified for that key are displayed in the Details pane. In the preceding display, the primary key consists of the **personid** column in the AUTHOR table.

The Associated Foreign Key Tables pane displays any foreign keys that are associated with a primary key or unique key that is selected in the Keys pane. The name of the foreign key and the name of the table that contains the foreign key are displayed. In the preceding display, the primary key AUTHOR.Primary is referenced by a foreign key in the BOOKS table.

The following display shows the **Keys** tab for the BOOKS table, the table that contains the foreign key that was referenced. The BOOKS table has two keys: a primary key named BOOKS.Primary and a foreign key named AUTHOR.BOOKS, which is selected on the left. Information about the foreign key is shown on the right.

**Display 4.4** Keys Tab with a Foreign Key Selected



The default name for a foreign key is **foreignTableName.currentTableName**, where **foreignTableName** is the name of the table where the foreign columns were originally created, and **currentTableName** is the name of the current table. In the preceding display, the foreign key is named AUTHOR.BOOKS, because the foreign columns originate in the AUTHOR table, and the current table is the BOOKS table.

When a foreign key is selected in the Keys pane, the following values are displayed in the Details pane:

- **Foreign Key Column** displays the column or combination of columns in the current table that references the corresponding column or combination of columns in another table. In the preceding display, the foreign key column is named **author**, which is the name of a column in the BOOKS table.
- **Length** displays the length of the Foreign Key Column.
- **Unique Key Column** displays the corresponding column or combination of columns in the other table. In the previous display, the unique key column is named **personid**.
- **Unique Key Table** displays the name of the other table: AUTHOR.

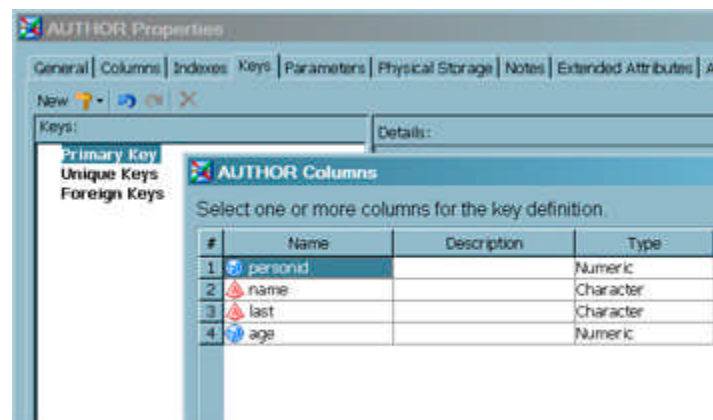


### Add a Primary Key or a Unique Key

In general, to create a primary key or a unique key, you select one or more columns in a table and specify them as a key. Typically, the creation of keys is carefully planned, so you know which table and columns to select. Perform the following steps to add a primary key or a unique key:

1. Access the **Keys** tab on the properties window for the desired table. For example, you want to create a primary key for the AUTHORS table.
2. Select **New** from the toolbar, and select **Primary Key** or **Unique Key**. Alternatively, right-click **Primary Key** or **Unique Key** in the Keys pane, and select **New**. A column selector window displays.
3. Select one or more columns in the current table that are appropriate for the key that you want to create. For example, the AUTHOR table has a column named **personid**, which uniquely identifies each author in the table. This is a good column to use as the primary key. The following display shows the selection of the **personid** column in the AUTHOR table.

**Display 4.5** Selecting a Column for a Primary Key

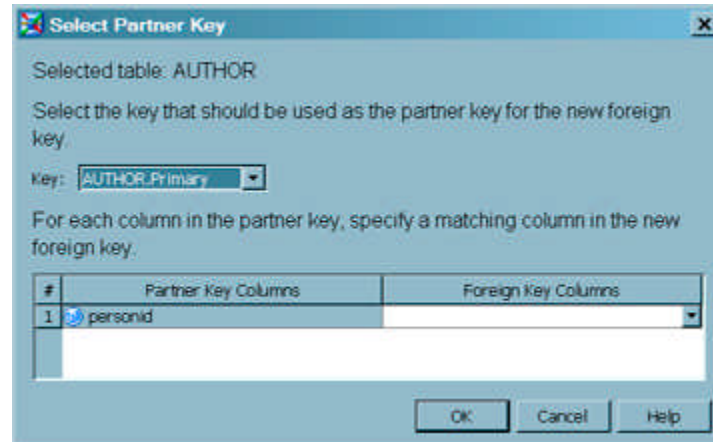


4. Click **OK** to save the selected columns in the metadata for a key. The new key is displayed in the Details pane.
5. Click **OK** to save the key.

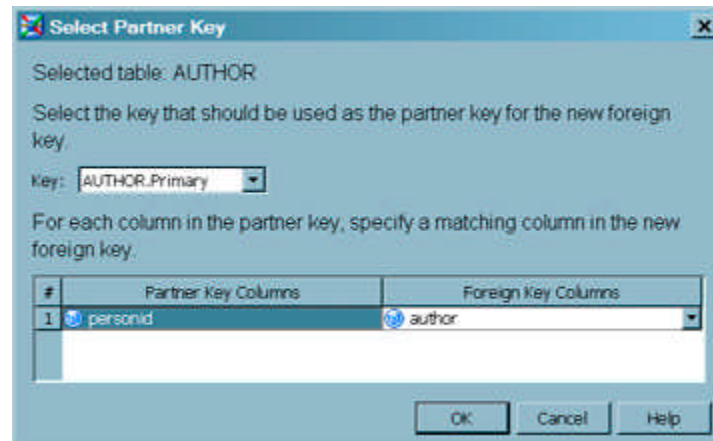
### Add a Foreign Key

To create a foreign key, which is a key in one table that references a corresponding key in another table, first select the other table that has the corresponding key. Then combine key columns in the current table with the corresponding key columns from the other table, and specify this combination as a foreign key. Typically, the creation of a foreign key is carefully planned, so you know which tables and columns to select. Perform the following steps to add a foreign key:

1. Access the **Keys** tab on the properties window for the table that requires a foreign key. For example, if you want to create a foreign key in the BOOKS table that references the primary key column in the AUTHORS table, then open the properties window for the BOOKS table.
2. Right-click **Foreign Key** in the Keys pane, and select **New**. A table selector window displays.
3. Select the other table with the column or columns that you want to reference in the current table. In the current example, select the AUTHORS table. Then, click **OK** to save your selection. The Select Partner Key window displays. A default partner column in the selected table is displayed in the **Partner Key Columns** field.

**Display 4.6** Foreign Key Column Not Yet Selected

4. If the default partner key column is not appropriate, use the **Key** selector to select a different key in the other table. Otherwise, accept the default. For example, in the preceding display, the default partner key column is the primary key column in the AUTHORS table: **personid**. You want to reference this column in the BOOKS table.
5. Use the selection arrow in the **Foreign Key Columns** field to select a column whose values should be linked to the partner key column. For example, the BOOKS table has a column named **author** whose values match the values in the **personid** column. The following display shows the combination of the **personid** column and the **author** column.

**Display 4.7** Foreign Key Column Selected

6. Click **OK** to save the selected columns in the metadata for the foreign key. The new key is displayed in the Details pane.
7. Click **OK** to save the key.

### **Update the Columns in a Key**

To add, delete, or change the order of columns in a primary key or unique key, select the key in the Keys pane, and then use the controls in the Details pane, such as the **Add** button, the up and down arrows, and so on. The only change you can make to a foreign key in the Details pane is to select a different foreign key column.

**Delete or Rename a Key**

To delete or rename a key, right-click the key in the Keys pane and select **Delete** or **Rename**.

*Note:* You cannot delete a primary key or a unique key that has a foreign key association. Deleting a key that is referenced by a foreign key breaks the table that contains the foreign key. You must delete the foreign key from the other table before you are permitted to delete the primary key or unique key in the current table.

---

## Maintaining Indexes

**Problem**

You want to create a new index for a table, or to modify or delete an existing index.

**Solution**

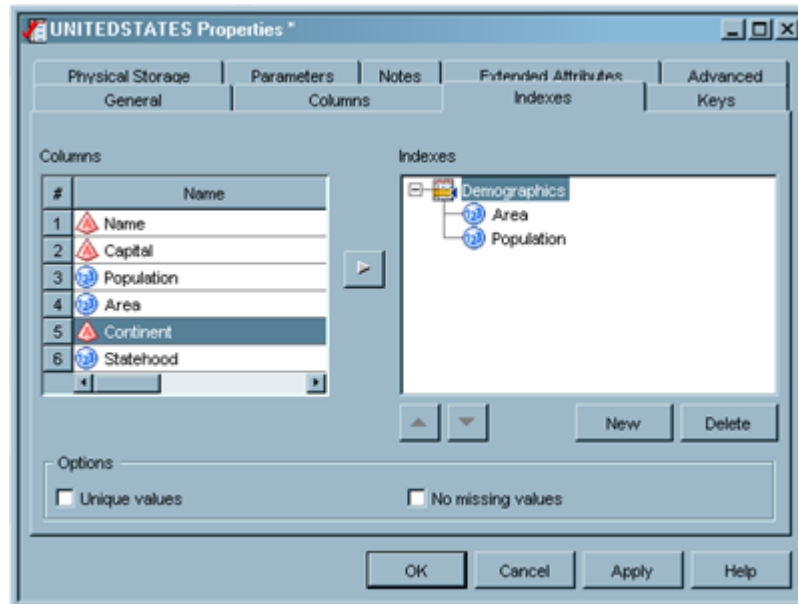
Use the **Indexes** tab on the properties window for the table to perform the following tasks:

- [“Create a New Index” on page 87](#)
- [“Delete an Index or a Column” on page 88](#)
- [“Rearrange the Columns in an Index” on page 88](#)

**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. Drag a column name from the **Columns** field to an index folder in the **Indexes** field to add one or more columns to the index.
3. Click **OK**. The following display depicts a sample index.

**Display 4.8** Sample Completed Index

*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**.

### **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 the **Indexes** 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**.

*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 on the current metadata server and exists in physical storage. You can browse temporary output tables until the Job Editor 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.

The View Data window constructs a SELECT query from the metadata for the selected table, view, external file, or transformation. For example, if the metadata for Table 1 specifies three columns that are named Col1, Col2, and Col3, then view data generates the following query for that table:

```
SELECT Col1, Col2, Col3 FROM Table1
```

If the metadata for a SAS or DBMS data store does not match the data in the data store, an error dialog box displays. The dialog box gives you the option of ignoring the column metadata that has been registered for the data store and using any column definitions in the data store to format the columns for display.

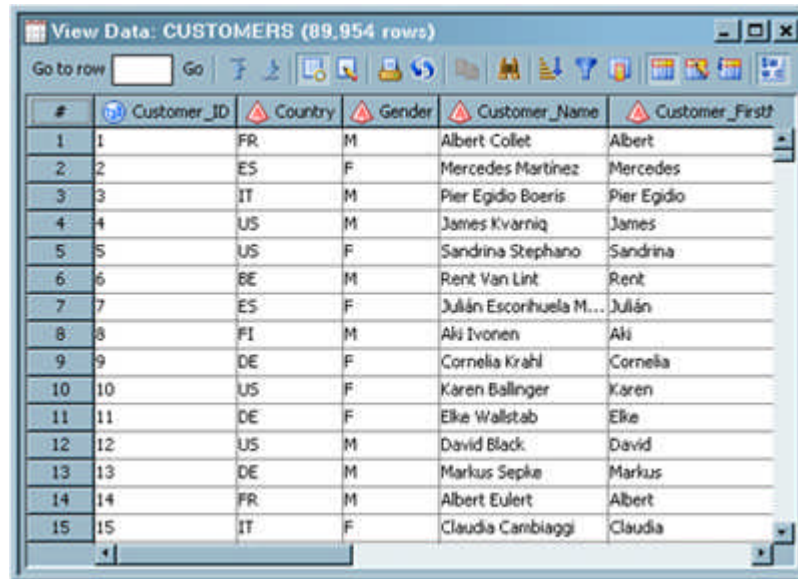
The View Data window cannot display data for a fixed-width external file unless the SAS informats in the metadata are appropriate for the columns in the data.

### **Tasks**

#### ***Use Browse Mode in the View Data Window***

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

1. Right-click the metadata object for the table, view, external file, temporary output, or transformation. Then, select **Open** 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 4.9** View Data Window in Browse Mode

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 4.10** View Data Browse Toolbar

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

**Table 4.8** Browse Functions in the View Data Window

Task	Action
Navigate within the data	<p>Perform the following steps:</p> <ul style="list-style-type: none"> <li>Enter a row number in the <b>Go to row</b> field and click <b>Go to row</b> to specify the number of the first row that is displayed in the table.</li> <li>Click <b>Go to first row</b> to navigate to the first row of data in the View Data window.</li> <li>Click <b>Go to last row</b> to navigate to the last row of data in the View Data window.</li> </ul>

Task	Action
Select a View Data window mode	<p>Perform the following steps:</p> <ul style="list-style-type: none"> <li>Click <b>Switch to browse mode</b> to switch to the browse mode.</li> <li>Click <b>Switch to edit mode</b> to switch to the edit mode.</li> </ul> <p>Note that the <b>Switch to browse mode</b> and <b>Switch to edit mode</b> buttons are displayed only for SAS tables.</p>
Perform utility functions	<p>Perform the following steps:</p> <ul style="list-style-type: none"> <li>Click <b>Print</b> to print the View Data window.</li> <li>Click <b>Refresh</b> to refresh the data in the View Data window.</li> </ul>
Copy one or more rows of data into the copy buffer	<p>Perform the following steps:</p> <ul style="list-style-type: none"> <li>Highlight one or more rows of data. Then, click <b>Copy</b> to copy the selected text into the copy buffer.</li> </ul>
Manipulate the data that is displayed in View Data window	<p>Perform the following steps:</p> <ul style="list-style-type: none"> <li>Click <b>Show search pane</b>. Then, use the search toolbar to search for string occurrences in the data set that is currently displayed in the View Data window.</li> <li>Click <b>Launch sort screen</b>. Then, use the <b>Sort By Columns</b> tab in the 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.</li> <li>Click <b>Filter</b>. Then, use the <b>Filter</b> tab in the 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.</li> <li>Click <b>Subset columns</b>. Use the <b>Columns</b> tab in the 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 that is currently displayed in the View Data window by selecting only some of the available columns in the <b>Columns</b> field. The redrawn View Data window includes only the columns that you select here on the <b>Columns</b> tab.</li> </ul>

Task	Action
Determine what is displayed in the column headings	<p>You can display any combination of column metadata, physical column names, and descriptions in the column headings.</p> <ul style="list-style-type: none"> <li>Click <b>Show column name in column header</b> to display physical column names in the column headings.</li> <li>Click <b>Show column description in column header</b> to display optional descriptions in the column headings.</li> <li>Click <b>Show column metadata name in column header</b> to display optional column metadata in the column headings. This metadata can be entered in some SAS Business Intelligence applications, such as the SAS Information Mapping Studio.</li> </ul>
Determine whether metadata formats are applied	<p>Perform the following steps:</p> <ul style="list-style-type: none"> <li>Click <b>Apply metadata formats</b> to toggle between showing formatted and unformatted data in the View Data window.</li> </ul>

To sort columns and perform related tasks, right-click on a column name and select an appropriate option from the pop-up menu. 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. For information about the available options, see [“Specifying Browse and Edit Options for Tables and External Files” on page 96](#).

---

## 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 SAS engine library and are assigned on the workspace server. 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. Right-click the metadata object for a SAS table. Then, select **Open** from the pop-up menu.



- 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.
- 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 4.11** View Data Window in Edit Mode

#	Customer_ID	Country	Gender	Customer_Name	Customer_I
1	1	FR	M	Albert Collet	Albert
2	2	ES	F	Mercedes Martinez	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	Julán Escorihuela Mo...	Julán
8	8	FI	M	Aki Iivonen	Aki
9	9	DE	F	Cornelia Krahl	Cornelia
10	9	DE	F	Cornelia Krahl	Cornelia
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.

- Double-click inside a cell and then change the data in the cell. Click **Save edit row** to commit the change to the database. Rows are committed as they are added. Of course, you must have operating system access for the file in order for the change to be saved.
- 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. If multiple rows have been deleted or pasted, then only the last row affected can be undone. Similarly, you can redo only your latest undo.
- Click a row number to select the row. Click **Copy** to copy the row into the buffer.
- Click **Go to last row** to move to the last row in the table.
- 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.

Note that you can also use **Paste Special** to paste more at once. You can copy single or multiple rows for pasting. When multiple rows are pasted, changes are made and the database table is immediately updated. If you paste a range of rows that go beyond that last row or if the range of the data is beyond the row and column range of the table, an error message is displayed. Use **Paste Special** to append new rows to the table by pasting data.

If you paste data into an EDIT row, only the first pasted row is considered. A warning to this effect is shown if more than one row is pasted. The pasted data is not automatically committed to the database.

- 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 4.12** View Data Edit Toolbar



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

**Table 4.9** Edit Functions in the View Data Window

Task	Action
Navigate within the data	Perform the following steps: <ul style="list-style-type: none"> <li>Enter a row number in the <b>Go to row</b> field and click <b>Go</b> to specify the number of the first row that is displayed in the table.</li> <li>Click <b>Go to first row</b> to navigate to the first row of data in the View Data window.</li> <li>Click <b>Go to last row</b> to navigate to the last row of data in the View Data window.</li> </ul>
Select a View Data window mode	Perform the following steps: <ul style="list-style-type: none"> <li>Click <b>Switch to browse mode</b> to switch to the browse mode.</li> <li>Click <b>Switch to edit mode</b> to switch to the edit mode.</li> </ul> Note that the <b>Switch to browse mode</b> and <b>Switch to edit mode</b> buttons are displayed only for SAS tables.
Perform utility functions	Perform the following steps: <ul style="list-style-type: none"> <li>Click <b>Print</b> to print the View Data window.</li> <li>Click <b>Refresh</b> to refresh the data in the View Data window.</li> </ul>
Copy or paste data	Perform the following steps: <ul style="list-style-type: none"> <li>Highlight one or more rows of data. Then, click <b>Copy</b> to copy the selected text into the copy buffer.</li> <li>Place the cursor in the row where you want to place the data. Then, click <b>Paste</b> to paste the data into the table. Note that you can also use <b>Paste Special</b> to paste more at once.</li> </ul>
Undo or redo editing operations	Perform the following steps: <ul style="list-style-type: none"> <li>Click <b>Undo last action</b> to reverse the most recent editing operation.</li> <li>Click <b>Redo last action</b> to restore the results of the most recent editing operation.</li> </ul>

Task	Action
Search the data displayed in View Data window	Perform the following steps: <ul style="list-style-type: none"> <li>Click <b>Show search pane</b>. Then, use the search toolbar to search for string occurrences in the data set that is currently displayed in the View Data window.</li> </ul>
Determine what is displayed in the column headings	You can display any combination of column metadata, physical column names, and descriptions in the column headings. <ul style="list-style-type: none"> <li>Click <b>Show column name in column header</b> to display physical column names in the column headings.</li> <li>Click <b>Show column description in column header</b> to display optional descriptions in the column headings.</li> </ul>
Commit or delete editing changes	Perform the following steps: <ul style="list-style-type: none"> <li>Click <b>Save edited row</b> to commit the changes that you have made to the currently edited row.</li> <li>Click <b>Delete selected rows</b> to delete the changes that you have made to the currently edited row.</li> </ul>

To hide, show, hold, and release columns, right-click on a column name and select an appropriate option from the pop-up menu.

To set options for the View Data window, select **Tool** ⇒ **Options** from the SAS Data Integration Studio menu bar to display the Options window. Then, click the **View Data** tab. For information about the available options, see [“Specifying Browse and Edit Options for Tables and External Files” on page 96](#).

---

## 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 New Table wizard.

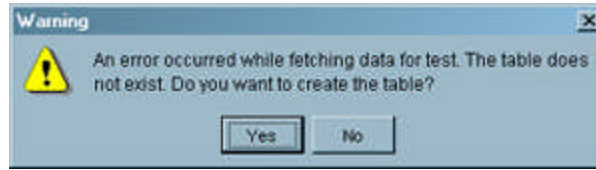
### Tasks

#### **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 New Table wizard. Select the columns that you need from existing tables.

- Right-click the newly registered table and click **Open**. The dialog box in the following display is shown.

**Display 4.13** Create Table Dialog Box

- 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.

---

## 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** menu is available on the **Tools** menu on the SAS Data Integration Studio menu bar.) 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 4.10** 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 Heading Options

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

**Table 4.11** Column Headings Options

Option	Description
Show column name in column header	Displays physical column names in the column headings.
Show column description in column header	Displays optional descriptions in the column headings.
Show column metadata name in column header	Displays optional column metadata names in the column headings. 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 headings 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 4.12** 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 4.13** 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 <b>Search for</b> 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 4.14** *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 causes 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	Default date values are returned as yyyy-MM-dd
Default datetime format	Default datetime values are returned as yyyy-MM-dd hh:mm:ss.SSS

## Chapter 5

# Working with External Files

---

<b>About External Files</b> .....	<b>100</b>
<b>Registering a Delimited External File</b> .....	<b>100</b>
Problem .....	100
Solution .....	100
Tasks .....	101
<b>Registering a Fixed-Width External File</b> .....	<b>103</b>
Problem .....	103
Solution .....	104
Tasks .....	104
<b>Registering an External File with User-Written Code</b> .....	<b>108</b>
Problem .....	108
Solution .....	108
Tasks .....	108
<b>Viewing or Updating External File Metadata</b> .....	<b>111</b>
Problem .....	111
Solution .....	111
<b>Overriding the Code Generated by the External File Wizards</b> .....	<b>112</b>
Problem .....	112
Solution .....	112
Tasks .....	112
<b>Specifying NLS Support for External Files</b> .....	<b>113</b>
Problem .....	113
Solution .....	113
Tasks .....	113
<b>Accessing an External File with an FTP Server or an HTTP Server</b> .....	<b>113</b>
Problem .....	113
Solution .....	114
Tasks .....	114
Additional Information .....	114
<b>Viewing Data in External Files</b> .....	<b>114</b>
Problem .....	114
Solution .....	115
Tasks .....	115
<b>Registering a COBOL Data File That Uses a COBOL Copybook</b> .....	<b>115</b>
Problem .....	115
Solution .....	116
Tasks .....	116

<b>Using an External File in the Process Flow for a Job</b> .....	<b>117</b>
Problem .....	117
Solution .....	117
Tasks .....	118

---

## About External Files

An external file, sometimes called a flat file or a raw data file, is a plain text file that often contains one record per line. Within each record, the fields can have a fixed length or they can be separated by delimiters, such as commas. Like SAS or DBMS tables, external files can be used as inputs and outputs in SAS Data Integration Studio jobs. Unlike SAS or DBMS tables, which are accessed with SAS LIBNAME engines, external files are accessed with SAS INFILE and FILE statements. Accordingly, external files have their own registration wizards, and they have two special transformations in the Transformations tree: File Reader and File Writer.

The most common tasks for external files are listed in the following table.

**Table 5.1** Common External File Tasks

Task	Action
Register an external file (add metadata about the file's physical location, columns, and other attributes).	For more information, see <a href="#">“Registering a Delimited External File” on page 100</a> , <a href="#">“Registering a Fixed-Width External File” on page 103</a> , and <a href="#">“Registering an External File with User-Written Code” on page 108</a> .
Specify a registered external file as a source or a target in a job.	For more information, see <a href="#">“Using an External File in the Process Flow for a Job” on page 117</a> .
View the data or metadata for a registered external file.	For more information, see <a href="#">“Viewing Data in External Files” on page 114</a> and <a href="#">“Viewing or Updating External File Metadata” on page 111</a> .

---

## 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 wizard to register the file. The wizard enables you to create metadata for external files that contain delimited data. This metadata is saved to a SAS Metadata Server.



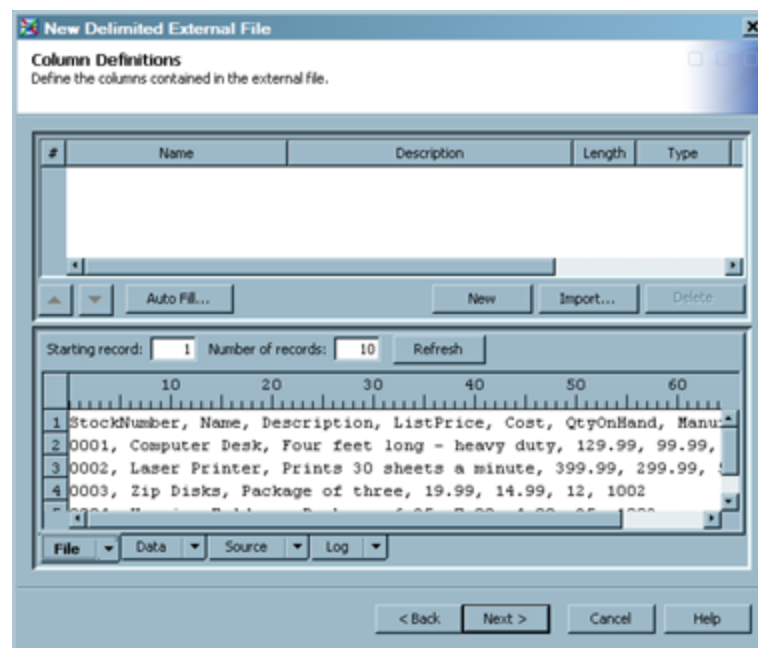
## Tasks

### Run the Delimited External File Wizard

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

1. Right-click the destination folder for the external file metadata. Then, select **New** ⇒ **External File** ⇒ **Delimited** to access the General page in the New User Written External File wizard. Enter an appropriate name and description of the external file that you want to register. Click **Next** to access the External File Location page.
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 page.
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 page.
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 page. Sample data is shown in the following display.

**Display 5.1** Sample Column Definitions



*Note:* If your external file contains fewer than 10 rows, a warning box is 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 page.
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 page.
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 page. 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 page 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 page.

**Display 5.2** Sample Completed Column Definitions Page

#	Name	Description	Length	Type
1	StockNumber		8	Numeric
2	Name		16	Character
3	Description		32	Character
4	ListPrice		8	Numeric

Starting record: 1 Number of records: 10 Refresh

1	StockNumber, Name, Description, ListPrice, Cost, QtyOnHand, Manu...
2	0001, Computer Desk, Four feet long - heavy duty, 129.99, 99.99,
3	0002, Laser Printer, Prints 30 sheets a minute, 399.99, 299.99,
4	0003, Zip Disks, Package of three, 19.99, 14.99, 12, 1002

File Data Source Log

< Back Next > Cancel Help

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 is properly displayed in the **Data** tab. The **Data** tab looks 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 is 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 is generated for the current external file when it is included in a SAS Data Integration Studio job.
13. Click **Next** and then **Finish** to save the metadata and exit the delimited external file wizard.

### 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 do not affect the physical data in the external file. However, the changes 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 following display.

**Display 5.3** Sample External File Columns Tab

#	Name	Description	Type	Length	Informat	Format	Is Nullable	Summary Role
1	StockNumber		Numeric	8,4	4		Yes	(None)
2	Name		Character	16,\$16	\$16		Yes	(None)
3	Description		Character	32,\$32	\$32		Yes	(None)
4	ListPrice		Numeric	8,7.2	7.2		Yes	(None)
5	Cost		Numeric	8,7.2	7.2		Yes	(None)
6	QtyOnHand		Numeric	8,3	3		Yes	(None)
7	ManufacturerID		Numeric	8,5	5		Yes	(None)

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

### View the Data

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

**Display 5.4** Sample External File Data in the View Data Window

#	StockNumber	Name	Description	ListPrice	Cost	QtyOnHand	ManufacturerID
1	1	Computer ...	Four feet long ...	129.99	99.99	5	1001
2	2	Laser Printer	Prints 30 sheets ...	299.99	299.99	5	1002
3	3	Zip Disks	Package of three	19.99	14.99	12	1002
4	4	Hanging Po...	Package of 25	7.99	4.99	25	1000
5	5	Calculator	Scientific with gr...	25.99	18.99	7	1004

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 **Open**.

## 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 wizard to register the file. The wizard enables you to create metadata for external files that contain fixed-width data. The metadata is saved to a SAS Metadata Server.

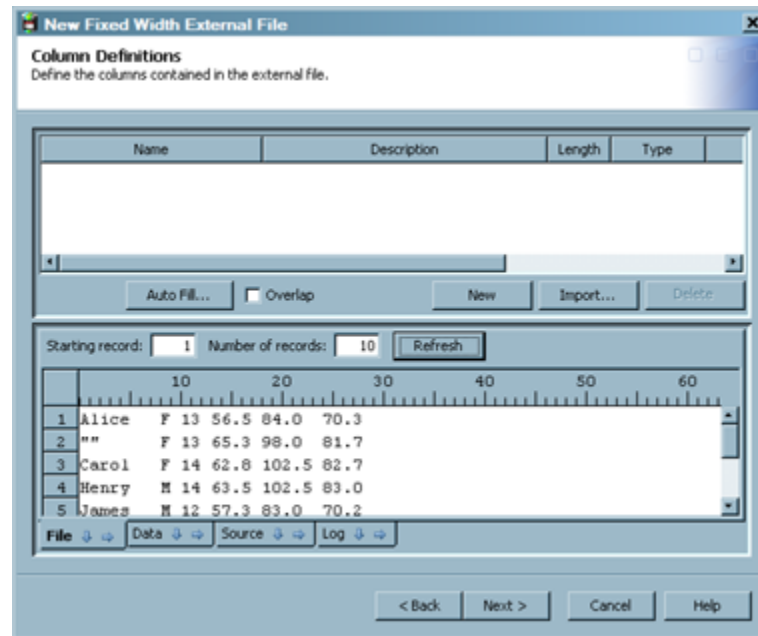
You need to know the width of each column in the external file. This information might be provided in a document that describes the structure of the external file.

## Tasks

### **Run the Fixed-Width External File Wizard**

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

1. Right-click the destination folder for the external file metadata. Then, select **New** ⇒ **External File** ⇒ **Fixed Width** to access the General page in the New Fixed Width External File wizard. Enter an appropriate name and description of the external file that you want to register. Click **Next** to access the External File Location page.
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 page.
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 page.
6. Click **Refresh** to view the raw data from the external file on the **File** tab in the view pane at the bottom of the page. Sample data is shown in the following display.

**Display 5.5** Sample Fixed-Width Data on the File Tab

7. 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 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. 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 page.
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 page. More accurate metadata is entered into the column components section of the page.
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 estimate of the data. You need to go in and verify this estimate. If appropriate informats are not provided for all variables in the fixed-width file, then incorrect results can be encountered when the external file is used in a job or when its data is viewed. A sample of a completed Column Definitions page is shown in the following display.

**Display 5.6** Sample Completed Column Definitions Page

**New Fixed Width External File**

**Column Definitions**  
Define the columns contained in the external file.

#	Name	Description	Length	Type
1	Student	Name of the student	7	Character
2	Gender	Gender of the student	3	Character
3	Age	Age of the student	8	Numeric
4	Midterms	Midterm examination score	8	Numeric

Auto Fill... ☐ Overlap New Import... Delete

Starting record: 1 Number of records: 10 Refresh

			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		

File Data Source Log

< Back Next > Cancel Help

You can click **Data** to see a formatted view of the external file data. To view the code that is 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 is generated for the current external file when it is included in a SAS Data Integration Studio job.

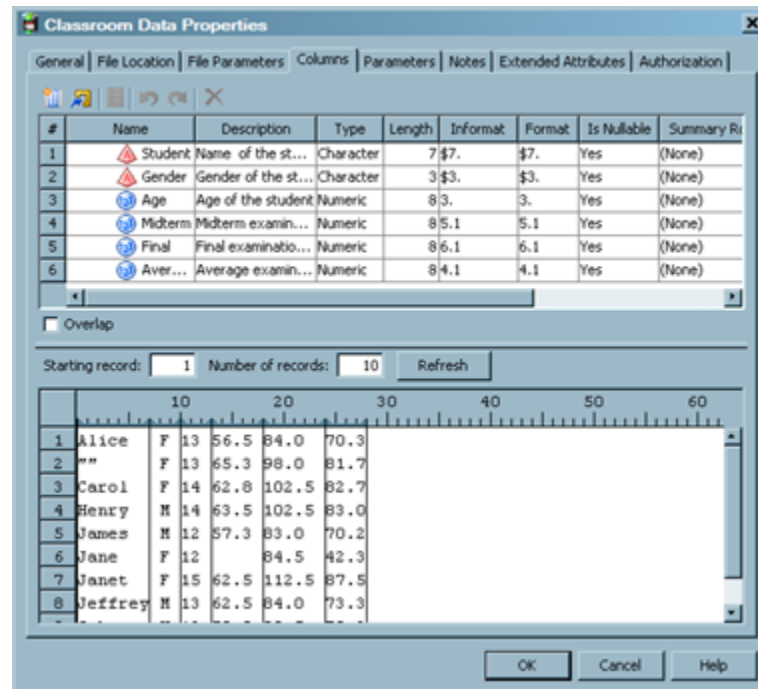
11. Click **Next** and **Finish** to save the metadata and exit the fixed-width external file wizard.

### 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 do not affect the physical data in the external file. However, the changes affect the data that is displayed 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 5.7 Sample External File Columns Tab



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

### View the Data

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

Display 5.8 Sample External File Data in the View Data Window

#	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 **Open**.

## Registering an External File with User-Written Code

### Problem

You want to register an external file whose structure is more complex than can be easily managed in the delimited wizard or the fixed width wizard.

### Solution

Use the New User-Written External File wizard to specify a user-written SAS INFILE statement to read the structure of the file. The wizard uses the INFILE statement to read the structure of the file, and then it registers the file on the metadata server. The metadata object for the file 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 wizard. 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 test your code:

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

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

    infile '\\machine number\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;

    Birthdayrun;
```



*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 the first two lines of code. They are not needed when the SAS code is used to process the external file.

3. Review the log in the Code 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 Code Editor window. Do not save the results.

### **Run the User-Written External File Wizard**

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

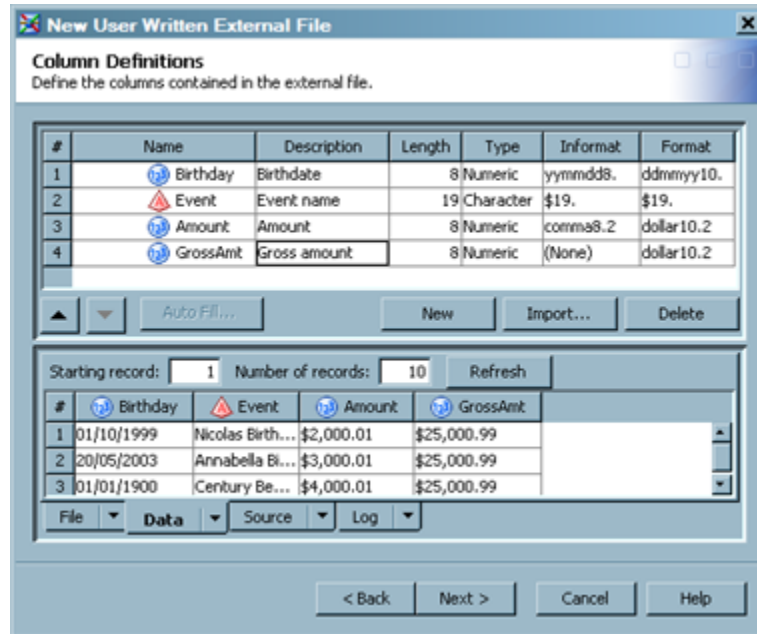
1. Right-click the destination folder for the external file metadata. Then, select **New** ⇒ **External File** ⇒ **User Written** to access the General page in the New Delimited External File wizard. Enter an appropriate name, description, and location of the external file that you want to register. Click **Next** to access the User Written Source Code page.
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**. For example, you can select **File** type to point to code that is embedded in a separate file. If you select **Metadata**, you must click **Edit** and paste the code in the **Edit Source Code** window.

*Note:* The **Host** and **Path** fields on the User Written Source Code page are displayed only when you select **File** in the **Type** field. Different fields are displayed when you select **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. For example, you can register the metadata for an external file that is named `birthday_event_data.txt`.
6. You can either enter the column definitions manually or click **Import** to access the Import Column Definitions window. For information about 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 **yyymmdd8.** 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 is 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 is 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.

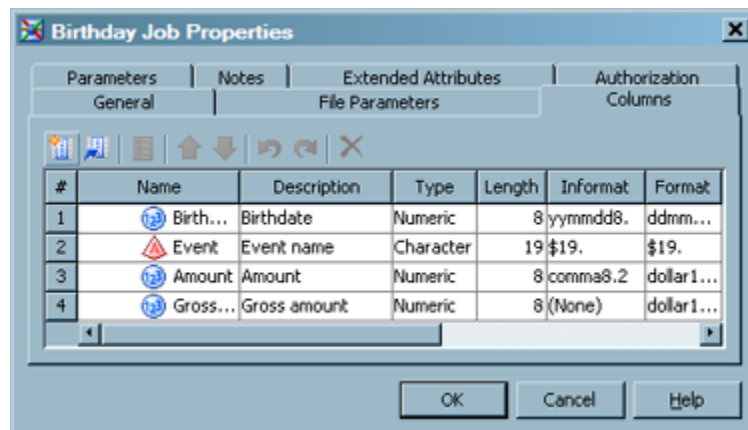
**Display 5.9** Completed Column Definitions Window

- Click **Next** to access a summary page, and then click **Finish** to save the metadata and exit the user written external file wizard.

### 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 do not affect the physical data in the external file. However, the changes 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:

- 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.

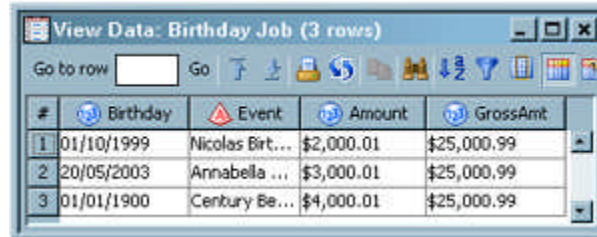
**Display 5.10** External File Columns Tab

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

### View the Data

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

**Display 5.11** 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 **Open**.

---

## 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
- Authorization

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 Job Editor 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 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.

---

## 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 external file wizard. To access this window, click **Advanced** on the Delimiters and Parameters page in the delimited external file wizard or on the Parameters page in the fixed-width external file 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 page, the **File** tab on the Columns Definition page, and the **Auto Fill** button on the Columns Definition page do 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. Right-click the destination folder for the external file metadata. Then, open the appropriate external file wizard and navigate to the Delimiters and Parameters page or the Parameters page (depending on the selected wizard).
2. Click the **Advanced** button to display the Advanced File Parameters window.
3. Select the **Override generated INFILE statement with the following statement** check box. Then, paste your SAS INFILE statement into the text area.
4. Enter other metadata for the external file as prompted by the wizard.

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 external file wizard.

### Tasks

#### **Specify NLS Encoding Options**

Perform the following steps to specify NLS encoding for the New Delimited External File wizard or the New Fixed Width External File wizard.

1. Right-click the destination folder for the external file metadata. Then, open the appropriate external file wizard. Enter appropriate settings on the General and External File Location pages. In particular, 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 does not work yet, because the required NLS options have not been specified.
2. Click **Next** to view either the Parameters page or the Parameters/Delimiters page, depending on the selected wizard.
3. Click **Advanced** to display the Advanced File Parameters window.
4. Enter the appropriate NLS encoding for the selected file in the **Encoding options** field. Then, click **OK**.

For detailed information about encoding values, see the section on "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 that server is registered to the current metadata server.

*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 page, the **File** tab on the Columns Definition page, and the **Auto Fill** button on the Columns Definition page do 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.

## 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. Right-click the destination folder for the external file metadata. Then, open the appropriate external file wizard and navigate to the External File Location page.
2. Click **Advanced**. 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 the section on "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 one of the external file wizards that are included in the wizard. 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 page or Columns Definition page 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 page 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 page in the external file wizards to view raw data for an unregistered file. You can also click the **File** tab on the Columns Definition page. There are two main situations where the **Preview** button and the **File** tab are not 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 [“Specifying NLS Support for External Files” on page 113.](#))
- 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 page in [“Registering a Delimited External File” on page 100.](#) You can also view the raw data in an external file after you have registered it in the wizard. To view the raw data, access the View File window for the external file. The raw data displayed in the external file wizards and the View File window is shown without detailed column specifications or data formatting. You can use the raw data to understand the structure of the external file better.

### View Formatted Data in the External File Wizards

The **Data** tab on the Columns Definition page 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 page in [“Registering a Delimited External File” on page 100.](#)

You can also view the formatted data in an external file after you have registered it in the wizard. To view the formatted data, 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 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 8:
  - a. Go to the Technical Support 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 is 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 filename:  
**MVS:wkyl.tst.v913.etls.copybook.**
- Native MVS includes partitioned data sets (PDS and PDSE). Take this factor 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:dwaterst.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.htm>.
- 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:dwaterst.tst.v913.cob2sas(r2cob1).**

### **Copy Column Metadata From the COBOL Format File**

Perform the following steps to copy column metadata from the COBOL format file in the Column Definitions page of an External File wizard.

1. Access the Column Definitions page of an External File wizard.
2. Click **Import** to access the Import Columns window.
3. Select the **Get the column definitions from a COBOL format file** 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 page.
4. Specify any remaining column metadata in the Column Definitions page. Click **Next**.
5. Click **Finish** when you are finished. The metadata for the external file is saved to the metadata server.

---

## **Using an External File in the Process Flow for a Job**

### **Problem**

You want the process flow for a job to read from an external file or write to an external file.

### **Solution**

In the process flow for a job, you can use the File Reader transformation to read an external file, and you can use the File Writer transformation to write to an external file.

An external file, sometimes called a flat file or a raw data file, is a plain text file that often contains one record per line. Within each record, the fields can have a fixed length or they can be separated by delimiters, such as commas. Most SAS Data Integration Studio transformations cannot use external files as inputs or outputs, so the File Reader and File Writer transformations are used to incorporate external files into the process flow for a job.

Perform the following tasks:

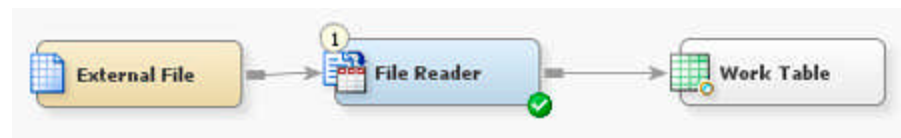
- “Read from an External File in a Job” on page 118
- “Write to an External File in a Job” on page 118
- “Run the Job and Verify the Results” on page 120

## Tasks

### **Read from an External File in a Job**

To read from an external file in a job, add a File Reader transformation to the job. Then, specify the external file as the input to the File Reader transformation, as shown in the next display.

**Display 5.12** File Reader Process Flow



The File Reader transformation reads information from the external file and writes the output to a temporary work table. By default, the temporary work table is a SAS data set. Most SAS Data Integration Studio transformations can read a SAS data set, so the output work table could be connected to a second transformation such as the Sort transformation. The second transformation could be connected to a third, and so on. In this way, a chain of transformations can be used to process information from an external file.

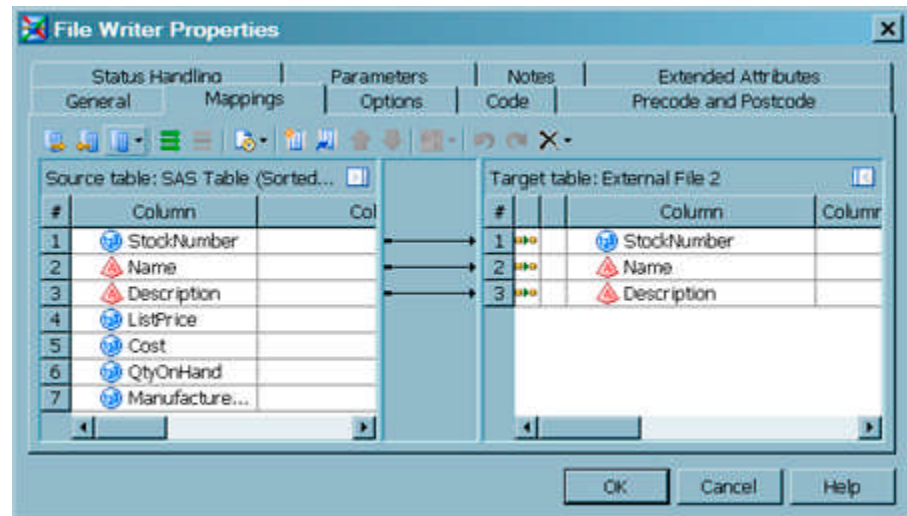
Perform the following steps to specify an external file as the input to the File Reader transformation.

1. If the external file has not been registered, use the appropriate wizard to register the external file. For more information, see “[Registering a Delimited External File](#)” on page 100, “[Registering a Fixed-Width External File](#)” on page 103, and “[Registering an External File with User-Written Code](#)” on page 108.
2. Create an empty SAS Data Integration Studio job. For more information, see “[Creating an Empty Job](#)” on page 123.
3. Select and drag a File Reader transformation from the Access folder of the Transformations tree. Then, drop it in the empty job on the **Diagram** tab in the Job Editor window.
4. Select and drag the external file from the tree view. Then, drop it before the File Reader transformation on the **Diagram** tab.
5. Drag the cursor from the external file to the input port of the File Reader transformation. This action connects the source to the transformation. At this point, the minimum process flow for your job should look similar to the preceding process flow. You can run the job and verify the results.

### **Write to an External File in a Job**

To write to an external file in a job, add a File Writer transformation to the job. Then, specify a SAS or DBMS table as the input and an external file as the output, as shown in the next display.



**Display 5.14** Mapping Tab for File Writer Transformation

In the preceding display, three columns from the input table (SAS Table) are mapped to three identical columns in the output file (External File 2). If the mappings are what you want, click **Cancel** to close the properties window. To update the mappings, see [“Maintaining Column Mappings” on page 154](#).

9. When ready, run the job and verify the results.

### **Run the Job and Verify the Results**

Perform the following steps to run the job and view the output.

1. Right-click on an empty area of the job, and click **Run** in the pop-up menu. SAS Data Integration Studio generates code for the job and submits it to the SAS Application Server for execution.
2. If error messages display, read and respond to the messages as needed.

Right-click the appropriate external file or table and select **Open** or **Open as Table** to verify that the correct data was loaded into the table or file.

## Chapter 6

# Creating Jobs

---

<b>About Jobs</b> .....	<b>122</b>
Jobs with Generated Source Code .....	122
Jobs with User-Supplied Source Code .....	123
Run Jobs .....	123
Manage Submitted Jobs .....	123
<b>Creating an Empty Job</b> .....	<b>123</b>
Problem .....	123
Solution .....	123
Tasks .....	123
<b>Creating a Process Flow for a Job</b> .....	<b>124</b>
Problem .....	124
Solution .....	124
Tasks .....	124
<b>Creating a Job That Contains Jobs</b> .....	<b>125</b>
Problem .....	125
Solution .....	125
Tasks .....	126
<b>Working with Default Temporary Output Tables</b> .....	<b>126</b>
Problem .....	126
Solution .....	126
Tasks .....	126
<b>About Job Options</b> .....	<b>130</b>
<b>Documenting Process Flow Diagrams</b> .....	<b>133</b>
Problem .....	133
Solution .....	133
Tasks .....	133
<b>Accessing Local and Remote Data</b> .....	<b>133</b>
Data Access Overview .....	133
Access Data in the Context of a Job .....	134
Access Data Interactively .....	134
Use a Data Transfer Transformation .....	135
<b>Viewing or Updating Job Metadata</b> .....	<b>136</b>
Problem .....	136
Solution .....	136
Tasks .....	136
<b>Displaying the SAS Code for a Job</b> .....	<b>137</b>
Problem .....	137

Solution .....	137
Tasks .....	137
<b>Common Code Generated for a Job .....</b>	<b>138</b>
Overview .....	138
LIBNAME Statements .....	138
SYSLAST Macro Statements .....	138
Remote Connection Statements .....	139
Macro Variables for Status Handling .....	140
User Credentials in Generated Code .....	140

---

## About Jobs

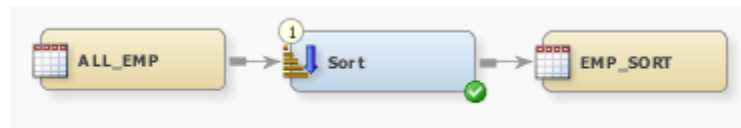
### *Jobs with Generated Source Code*

A job is a collection of SAS tasks that create output. SAS Data Integration Studio uses the metadata for each job to generate 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 6.1** *Process Flow Diagram for a Job That Sorts Data*



The components of this process flow perform the following functions:

- ALL\_EMP specifies metadata for the source table.
- Sort specifies metadata for the sort process.
- EMP\_SORT specifies metadata for the target table.

SAS Data Integration Studio uses this metadata to generate SAS code that reads ALL\_EMP, sorts this information, and then writes it to the EMP\_SORT table. You can also include temporary output tables and Table Loader transformations in process flows. For information, see [“Working with Default Temporary Output Tables” on page 126](#).

Each process in a process flow diagram is specified by a metadata object called a transformation. In the example, SAS Sort is a transformation. 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 in the preceding example, see [“Creating a Process Flow for a Job” on page 124](#).

## **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 [“About User-Written Code” on page 213](#).

## **Run Jobs**

There are four ways to run a job:

- submit the job for immediate execution. For information, see [“Submitting a Job for Immediate Execution” on page 142](#).
- deploy the job for scheduling. For information, see [“Deploying Jobs for Scheduling” on page 185](#).
- deploy the job as a SAS stored process. For information, see [“Deploying Jobs as Stored Processes” on page 190](#).
- deploy a stored process as a Web service. For information, see [“Deploying a Stored Process as a Web Service” on page 203](#).

## **Manage Submitted Jobs**

After you have submitted the job, you can use the tabs in the Details panel to check status, review warnings and errors, examine statistics, and trace the control flow of the job. For details, see [“About Managing Jobs” on page 142](#).

*Note:* You can also trace the control flow of a job before you run the job.

---

# **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 Job Editor window.

## **Solution**

Use the New Job wizard to create an empty job in a specified location.

## **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:
  - Select **File** ⇒ **New** from the menubar. Then, click **Job**.
  - Click **New** on the toolbar. Then, click **Job**.

- Right-click on the folder where you want the job to be located and click **New**. Then, click **Job**.
2. Enter an appropriate name for the job in the New Job wizard in the **Name** field. You can enter an optional description of the job in the **Description** field. You can also browse for a location for the job's metadata by using the **Browse** button and the **Location** field.
  3. Click **OK** to save the job.

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 that populates a target table with data. Then, you need to populate the job with the source tables, transformations, and target tables that are 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 Job Editor window with the source tables, transformations, and target tables that you need to accomplish your task. Note that some transformations do not support permanent target tables.

### Tasks

#### Create and Populate a Sample Job

Perform the following steps to create and populate a job:

1. Create an empty job. For information, see [“Creating an Empty Job” on page 123](#).
2. Drop the source table on the **Diagram** tab of the Job Editor window. Sources must be registered in SAS Data Integration Studio. You can also right-click a source table (or any object that can be dropped into a job) in an Inventory tree and click **Add to Diagram** in the pop-up menu. This action adds the selected object to the **Diagram** tab of the active job on the desktop. Of course, this option is available only when at least one job is open.
3. Drop a transformation from the Transformations tree on the **Diagram** tab.
4. Drag the cursor from the source table to the input port of the transformation. This action connects the source to the transformation. If the input port that you need is not available, right-click the transformation and click **Ports** in the pop-up menu. Then, click **Add Input Port** in the sub-menu. This feature is available for most transformations. It enables you to perform the following tasks:
  - Add an input port.
  - Delete an input port.
  - Add an output port.
  - Delete an output port.

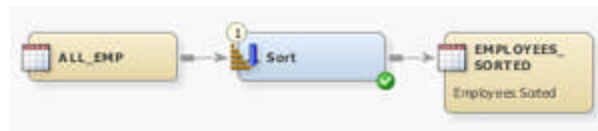


*Note:* You can include a particular table more than once in a process flow. For example, you can use the same table as the source table and the target table for a SAS Data Integration Studio job. You can use this approach to change the structure of a physical table. However, the control flow tab might not report control flow warnings correctly if you do this.

5. Because you want to have a permanent target table to contain the output for the transformation, right-click the temporary work table that is attached to the transformation, click **Replace** in the pop-up menu. Then, use the Table Selector window to select the target table for the job. The target table must be registered in SAS Data Integration Studio. (For more information about temporary work tables, see [“Working with Default Temporary Output Tables”](#) on page 126.)

The following display shows a process flow diagram for a sample job that contains the Sort transformation.

**Display 6.2** Sample Process Flow



Note the source table is named ALL\_EMP and target table is named EMPLOYEES\_SORTED.

You can set global options for jobs on the **Code Generation** tab of the **Options** menu. The Options window is available from the **Tools** menu on the SAS Data Integration Studio menu bar. You can set local options on the **Options** tab that is on the properties window for each table. For detailed information, see [“About Job Options”](#) on page 130.

*Note:* If you change a job in any way, you must save the job in order to save the changes. You should save the whole job even when you click **Save** or **Save As** on the **Code** tab for a job or transformation or the **Precode and Postcode** tab for a transformation in a job. These save options save the updated code to the metadata or to a file, but the link between the saved code and the job is not established unless the job is saved.

---

## Creating a Job That Contains Jobs

### Problem

You want to create a job that contains one or more existing jobs.

### Solution

You can add existing jobs from a tree view to the **Diagram** tab of the Job Editor window in an open job. These jobs are added to the control flow in the order that they are added to the job. This sequence is useful for jobs that are closely related, but the jobs do not have to be related. You can always change the order of execution for the added jobs in the **Control Flow** tab of the Details pane.

## Tasks

### Create a Job That Contains Existing Jobs

Perform the following tasks to create a job that contains existing jobs:

1. Create an empty SAS Data Integration Studio job.
2. Drag one or more existing jobs from a tree view to the **Diagram** tab of the Job Editor window. The completed sample job is shown in the following display.

**Display 6.3** Completed Job



Note that the added jobs are linked by dashed-line control flow arrows and not by solid-line data flow arrows. By default, the extract job in the sample job, which was added first, will be executed first. Then the sort job, which was added second, will be executed.

---

## Working with Default Temporary Output Tables

### Problem

You added a transformation to the **Diagram** tab of the Job Editor window. The transformation sends its output to a temporary output table, and you need to decide what you should do with the temporary output table. Of course, the temporary output table is populated with data only when the job that contains it has been run.

### Solution

You can use default temporary output tables in the following ways:

- [“Use the Default Temporary Output Table As the Final Output” on page 126](#)
- [“Use the Default Temporary Output Table As an Input to Another Transformation” on page 127](#)
- [“Replace the Default Temporary Output Table with a Permanent Target Table” on page 128](#)
- [“Use the Temporary Output Table As an Input to a Table Loader ” on page 129](#)

## Tasks

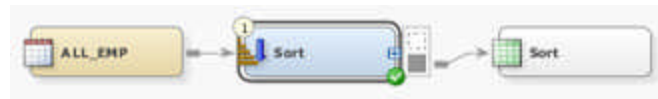
### Use the Default Temporary Output Table As the Final Output

When the default temporary output table is placed at the end of a job, you can keep the table and use it to view the output of the transformation. Then, you can review the results of the transformation without writing the data to a permanent target table. Perform the following steps to create a process flow diagram that uses the default temporary output table as the final output:

1. Create an empty job.
2. Select and drag a transformation from the Transformations tree. Then, drop it in the empty job on the **Diagram** tab in the Job Editor window.
3. Select and drag a source table from the Inventory tree. Then, drop it before the transformation on the **Diagram** tab.
4. Drag the cursor from the source table to the input port of the transformation. This action connects the source to the transformation.

The following display shows a sample job that works this way.

**Display 6.4** Sample Job with Default Temporary Output Table



By default, the temporary output table for single-output transformations has the same name as the transformation that provides its input. However, when a transformation has multiple outputs, a numerical suffix is added to each output table (for example, Splitter 0 and Splitter 1). In addition, users can change these default names in the property window for the table. The new name must be a valid SAS table name, just like the name for any other table.

### ***Use the Default Temporary Output Table As an Input to Another Transformation***

You can't use one transformation as the direct data input to another transformation. The data must first flow from a transformation to a permanent or temporary output table. Then, it can proceed to the next transformation.

Of course, if you need to save the output into a physical table that you can access after the current SAS session is terminated, you must use a permanent output table. You need to consider performance when you decide whether to use permanent or temporary output storage.

Temporary output storage can be created either as a table in the WORK library or as a view. If the data from the first transformation in the job is referenced multiple times in a process flow, then putting the data into a table generally improves overall performance. When you use a view as a temporary output table, SAS must execute the underlying code repeatedly each time the view is accessed.

However, if the data is referenced only once in a process flow, then the use of a view that is created from a temporary output table usually offers better performance.

You can tell whether a temporary output table takes the form of a view or a physical table by looking for the View modifier on the temporary output table. You can also right-click a temporary output table and look at the pop-up menu. If the **Create as View** item is checked, a view is generated. If not, the output is stored in a temporary physical table.

You can also click **Create as View** to switch between a physical table and a view. Note, however, that some transformations, such as Sort, do not support the creation of views. You can click **Create as View**, but the transformation ignores it and produces a temporary physical table.

Perform the following steps to create a process flow diagram that uses a temporary output table as an input to a transformation:

1. Create an empty job.

2. Select and drag a transformation from the Transformations tree. Then, drop it in the empty job on the **Diagram** tab in the Job Editor window.
3. Select and drag a source table from the Inventory tree. Then, drop it before the transformation on the **Diagram** tab.
4. Drag the cursor from the source table to the input port of the transformation. This action connects the source to the transformation.
5. Select and drag a second transformation from the Transformations tree on the **Diagram** tab.
6. Drag the cursor from the output port of the temporary output table that is attached to the first transformation to the input port of the second transformation. This action connects the temporary output table to the second transformation.

The following display shows a sample job that works this way.

**Display 6.5** Sample Job with Default Temporary Output Table between Transformations



*Note:* Some transformations, such as Return Code Check, produce no data output. Because they are not data transformations, they are linked to other transformations only by control flow lines. The User Written transformation also has an optional data target. When it is used without a data target, it also connects only with control flow lines.

### **Replace the Default Temporary Output Table with a Permanent Target Table**

You can replace the default temporary output table with a permanent target table. Then, you can write the data directly to the target table without first passing it through a temporary view. You might use this approach with the last transformation in a process flow, which is when you want to store the output in a permanent table. These permanent target tables perform better than temporary output tables under the following conditions:

- The data is referenced multiple times in a process flow. In a temporary output table, SAS must execute the underlying code repeatedly each time the view is accessed.
- The data is referenced once in a process flow, but the reference is a resource-intensive procedure that performs multiple passes of the input.
- The data is generated with SQL and is referenced once, but the reference is from another SQL view. SAS SQL optimization can be less effective when views are nested. This is especially true if the steps involve joins or RDBMS sources.

Note that these performance issues occur when the temporary output table takes the form of a view.

Perform the following steps to create a process flow diagram that replaces the default temporary output table with a permanent table:

1. Create an empty job.
2. Select and drag a transformation from the Transformations tree. Then, drop it in the empty job on the **Diagram** tab in the Job Editor window.
3. Select and drag a source table from the Inventory tree. Then, drop it before the transformation on the **Diagram** tab.

4. Drag the cursor from the source table to the input port of the transformation. This action connects the source to the transformation.
5. Right-click the temporary output table that is attached to the transformation. Then, click either **Register Table** or **Replace** in the pop-up menu.

- Click **Register Table** to display a Register Table window that enables you to change the temporary output table into a permanent physical table. This permanent table is displayed on the **Diagram** tab of the Job Editor window and added to the Inventory tree.

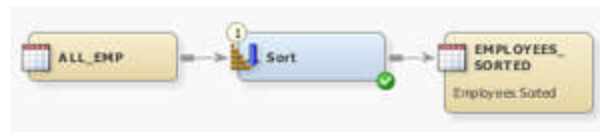
The table is added to the library that was used when the register table function was last run in the current SAS session. If register table has not been used in the current session, then you must add a library for the table on the **Physical Storage** tab of the Register Table window. This step prevents a design-time warning in the Job Editor.

- Click **Replace** to display a Table Selector window that enables you to replace the selected temporary output table with a specified physical table. If you want to retain the mappings, then choose a physical table that matches the temporary table.

Both the register table and replace functions attempt to keep mappings and expressions intact. When you simply delete the temporary table and connect the transformation directly to a target table that you drop on the **Diagram** tab, these mappings are lost.

The following display shows a sample job that includes a permanent target table.

**Display 6.6** Sample Job with a Permanent Target Table



### ***Use the Temporary Output Table As an Input to a Table Loader***

You can always let a SAS Data Integration Studio transformation perform a simple load of its output table that drops and replaces the table. However, you can also add a Table Loader transformation to a permanent output table. Then, you can use the options in the Table Load transformation to control how data is loaded into the target table. In fact, a separate Table Loader transformation might be desirable under the following conditions:

- loading a DBMS table with any technique other than drop and replace
- loading tables that contain rows that must be updated upon load (instead of dropping and recreating the table each time the job is executed)
- creating primary keys, foreign keys, or column constraints
- performing operations on constraints before or after the loading of the output table
- performing operations on indexes other than after the loading of the output table

Note that some of these actions are also possible with the SCD Type 2 Loader transformation.

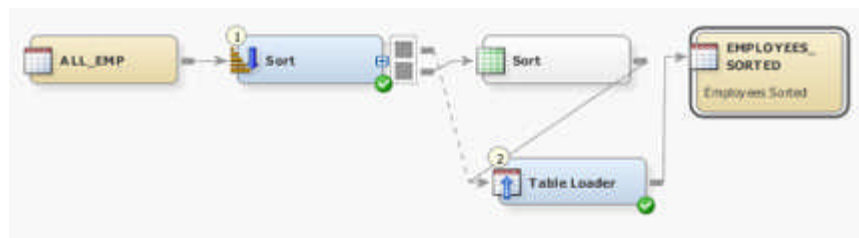
Perform the following steps to create a sample process flow diagram that includes a source table, an initial transformation, a temporary output table, a Table Loader transformation, and a permanent target table:

1. Create an empty job.

2. Select and drag a transformation from the Transformations tree. Then, drop it in the empty job on the **Diagram** tab in the Job Editor window.
3. Select and drag a source table from the Inventory tree. Then, drop it before the transformation on the **Diagram** tab.
4. Drag the cursor from the source table to the input port of the transformation. This action connects the source to the transformation.
5. Select and drag a Table Loader transformation from the Transformations tree on the **Diagram** tab.
6. Drag the cursor from the output port of the temporary output table that is attached to the first transformation to the input port of the Table Loader transformation. This action connects the temporary output table to the Table Loader transformation.
7. Select and drag the target table out of the Inventory tree. Then, drop it after the Table Loader transformation on the **Diagram** tab.
8. Drag the cursor from the output port of the Table Loader transformation to the input port of the target table. This action connects the Table Loader transformation to the target table.

The following display shows a sample job that works this way.

**Display 6.7** Sample Job with a Default Temporary Output Table and a Table Loader



You can feed any table, temporary output table, or physical table into a Table Loader transformation. For example, you can omit the initial Sort transformation and its input and output tables. Then, the job consists of a table that feeds into the Table Loader transformation. The Table Loader then feeds into the target table. In fact, you can use the same table as both the input and the output for the Table Loader, as shown in the following display.

**Display 6.8** Sample Job Table Loader and a Single Table



This approach enables you to use the Table Loader transformation to reload the table with a different load technique.

## About Job Options

Options can be set for SAS 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** ⇒ **Code Generation** 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 6.1** Global Options for Jobs

Option Name	Description
Enable optional macro variables for new jobs	When selected, specifies that optional metadata macro variables are to be included in the code that SAS Data Integration Studio generates for new jobs.
Generate JCL compatible code with an extra space at beginning of each line	When selected, generates JCL compatible code with an extra space at the beginning of each line.
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 specific applications to run on designated servers. The grid workload specification consists of a string value that must match the name that is defined in the Platform Computing grid configuration files. These files are text files that are 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.

Option Name	Description
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.
Enable generation of job status handling as last step for new jobs	When selected, adds job status handling code as the last step for new jobs.
Use locale-specific date format in generated code for new jobs	When selected, uses the appropriate date format for the specific locale in the generated code for new 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.

**Table 6.2** Local Options for Jobs

Option name	Description
System Options	Enables you to set options by using a SAS OPTIONS statement.
Enable optional metadata 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.
Generate the status handling code as the last step	When selected, adds job status handling code as the last step.
Use locale-specific date format in generated code	When selected, uses the appropriate date format for the specific locale in the generated code.



---

## Documenting Process Flow Diagrams

### **Problem**

You want to document a process flow diagram by either printing it directly or saving it as a graphic file. The diagram has been built on the **Diagram** tab in the Job Editor window of a SAS Data Integration Studio job.

### **Solution**

You can print or save the process flow diagram from the Job Editor window of an open job.

### **Tasks**

#### ***Print or Save a Process Flow Diagram***

Perform the following steps to print or save a process flow diagram:

1. Locate and open the job that contains the process flow diagram that you need to document.
2. If you want to print the process flow diagram, select **File** ⇒ **Print** from the menu bar. The Print window displays. Then, configure and run the print job. Note that the process flow diagram is resized to fit the paper that is selected for the printer. Use a plotter for large process flow diagrams.
3. If you want to print the process flow diagram as a graphic file, select **File** ⇒ **Save Diagram as Image** from the menu bar. A submenu displays the following two options: **Current Page** or **Entire Diagram**. The **Entire Diagram** option allows the user to save the entire image, but it is scaled and might lose some resolution for extremely large images. The **Current Page** option creates an image of the visible portion of the flow without scaling. After selecting an option, specify a name and path and click **Save** to save the file.

---

## Accessing Local and Remote Data

### ***Data Access Overview***

You can access data using the following methods:

- [“Access Data in the Context of a Job” on page 134](#)
- [“Access Data Interactively” on page 134](#)
- [“Use a Data Transfer Transformation” on page 135](#)

## Access Data in the Context of a Job

You can access data implicitly in the context of a job. 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 run in each iteration of the parallel process.

For information about accessing remote data in the context of a job, administrators should see the section on "Multi-Tier Environments" in the "SAS Data Integration Studio" chapter of the *SAS Intelligence Platform: Desktop Application Administration Guide*.

Administrators should also see [“Using Deploy for Scheduling to Execute Jobs on a Remote Host” on page 188](#). 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 138](#).

## 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 external file wizards, 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 **Open** 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 the section on "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 also 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.

---

## 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, then the job must also have a process flow diagram. If you supply the source code for a job, then 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. Find the job on the SAS Data Integration Studio desktop. Common job locations include the following:
  - the Jobs folder in the Inventory tree
  - the My Folder folder
  - the Shared Data folder
  - a folder nested in the User folder
2. Right-click the desired job. Then, click **Properties** in the pop-up menu 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. Locate the job.
2. Open the job by using one of the following methods:
  - Double click the job.
  - Right-click the job. Then, click **Open** in the pop-up menu.Both methods display the process flow diagram for the job in the **Diagram** tab in the Job Editor window.
3. View or update the metadata displayed in the process flow diagram by using one of the following methods:

- To update the metadata for tables or external files in the job, see [“Viewing or Updating Table Metadata” on page 68](#) or [“Viewing or Updating External File Metadata” on page 111](#).
- To update the metadata for transformations in the job, see [“Viewing or Updating the Metadata for Transformations” on page 165](#).
- To add a transformation to a process flow diagram, select the transformation and drop it in the Job Editor 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 142](#).

---

## 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 213](#).)

### Solution

You can display the SAS code for a job on the **Code** tab of the Job Editor window or on the **Code** tab of a job properties window. In either case, 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. See [“Connecting to a SAS Metadata Server” on page 26](#).

### Tasks

#### **View SAS Code in the Code Tab of a Job Editor Window**

You can view the code for a job that is currently displayed in the Job Editor window. To do this, click the **Code** 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 on the **Code** tab.

#### **View SAS Code on the Code Tab in the Job Properties Window**

Perform the following steps to view the code for a job that is not displayed in the Job Editor 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 **Properties** from the pop-up menu.
3. Click the **Code** tab in the properties window to review the code.
4. Click **OK** to close the properties window.

## Common Code Generated for a Job

### Overview

When SAS Data Integration Studio generates code for a job, it typically generates the following items:

- “LIBNAME Statements” on page 138
- “SYSLAST Macro Statements” on page 138
- “Remote Connection Statements” on page 139
- “Macro Variables for Status Handling” on page 140
- “User Credentials in Generated Code” on page 140

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 “About Job Options” on page 130.

### 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=database>;
```

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;
```

### 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*

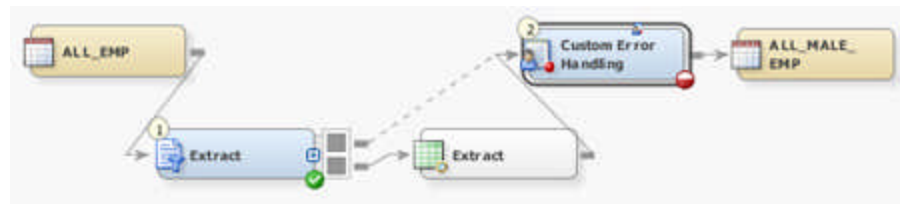
is the name of the last output table created by the transformation. The SYSLAST macro variable is used to make

*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 if that transformation does not produce output, or if 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 6.9** Process Flow with a Custom Error Handling Transformation



In this example, the Custom Error Handling transformation contains user-written code that handles errors from the Extract transformation, and the error-handling code does not produce output that should 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. The Custom Error Handling transformation was created with the User Written Code transformation. This particular instance of the transformation was renamed to Custom Error Handling.

In this example, 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 Handling 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 [“About Status Handling for Jobs and Transformations” on page 167](#).

### ***User Credentials in Generated Code***

The code that is generated for a job contains the credentials of the user who created the job. If a user'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 Deploying Jobs for Scheduling” on page 185](#).



## Chapter 7

# Managing Jobs

---

<b>About Managing Jobs</b>	<b>142</b>
<b>Submitting a Job for Immediate Execution</b>	<b>142</b>
Problem	142
Solution	142
Tasks	143
<b>Meeting Prerequisites for Collecting Job Statistics</b>	<b>145</b>
<b>Reviewing a Successful Job</b>	<b>145</b>
Problem	145
Solution	145
Tasks	145
<b>Diagnosing and Correcting an Unsuccessful Job</b>	<b>150</b>
Problem	150
Solution	150
Tasks	150
<b>Maintaining Column Mappings</b>	<b>154</b>
Problem	154
Solution	154
Tasks	154
<b>Managing the Scope of Column Changes in Jobs</b>	<b>158</b>
Problem	158
Solution	158
Tasks	158
<b>Managing Connections in Job Editor Windows</b>	<b>162</b>
Problem	162
Solution	162
Tasks	162
<b>Viewing the Code for a Transformation</b>	<b>164</b>
Problem	164
Solution	164
Tasks	164
<b>Viewing or Updating the Metadata for Transformations</b>	<b>165</b>
Problem	165
Solution	165
Tasks	165

---

## About Managing Jobs

Once you have created a SAS Data Integration Studio job, you need to be able to run it, check its status, review warnings and errors, examine statistics, and trace the control flow of the job. These job management practices are covered in the following topics:

- [“Submitting a Job for Immediate Execution” on page 142](#)
- [“Meeting Prerequisites for Collecting Job Statistics” on page 145](#)
- [“Reviewing a Successful Job” on page 145](#)
- [“Diagnosing and Correcting an Unsuccessful Job” on page 150](#)
- [“Maintaining Column Mappings” on page 154](#)
- [“Managing the Scope of Column Changes in Jobs” on page 158](#)
- [“Managing Connections in Job Editor Windows” on page 162](#)

---

## Submitting a Job for Immediate Execution

### **Problem**

You want to execute a job immediately.

### **Solution**

You can submit a job from the Job Editor window 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. Perform the following tasks:

- [“Submit a Complete Job” on page 143](#)
- [“Submit Selected Transformations in a Job” on page 143](#)
- [“Submit a Segment of a Job” on page 144](#)
- [“Submit a Job One Step at a Time” on page 144](#)
- [“Submit a Job to a Grid” on page 144](#)

*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 be installed, and the appropriate metadata must be defined for it. For details, see [“Selecting a Default SAS Application Server” on page 30](#).

## Tasks

### Submit a Complete Job

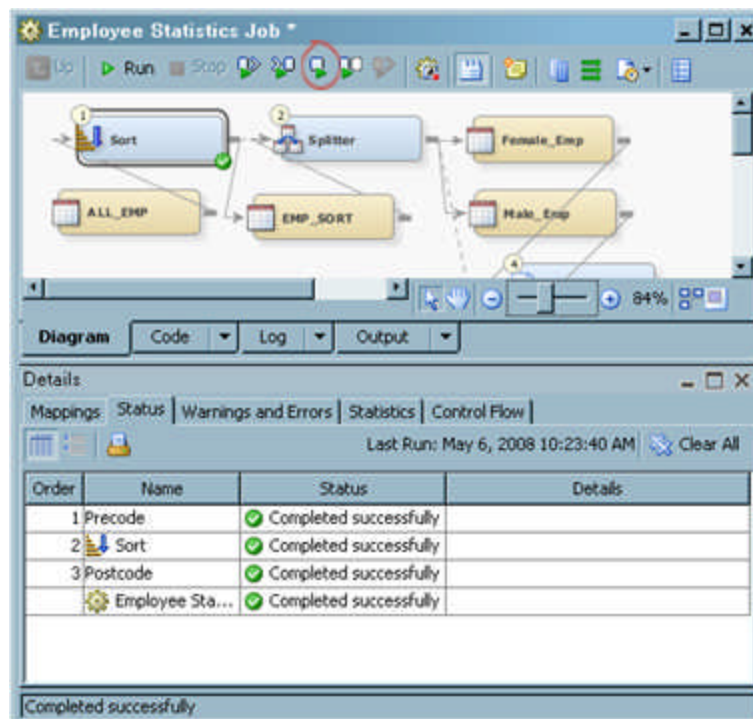
You can submit a job that is displayed in a Job Editor window. Click **Run** on the toolbar for the job, or right-click on a blank space in the job and click **Run** in the pop-up menu. The job is submitted to the default SAS Application Server and to any server that is specified in the metadata for a transformation within the job.

### Submit Selected Transformations in a Job

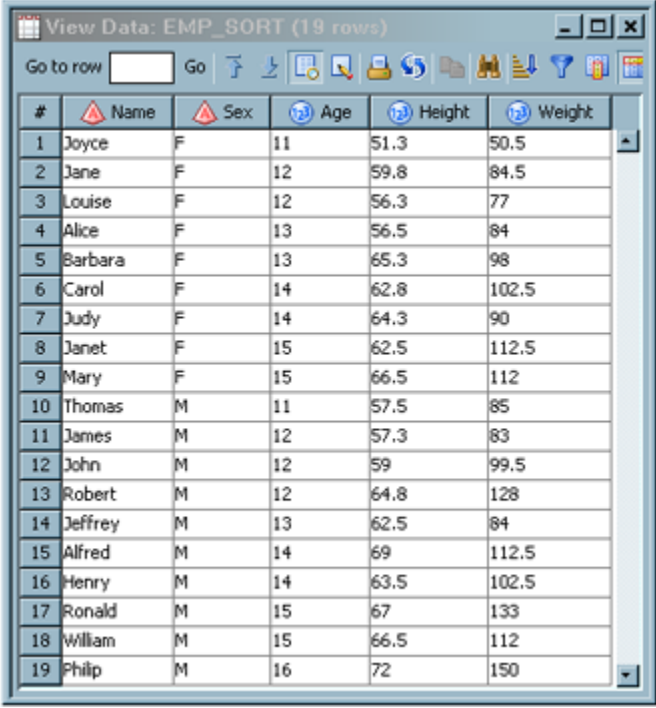
You can submit selected transformations in a job that is displayed in a Job Editor window. This function enables you to submit a portion of a job without submitting the entire job. For example, you can re-sort a long job without consuming the resources that are required if you submit the entire job. Perform the following steps to submit selected transformations in a job:

1. Control-click the transformations that you want to submit for execution. (You can simply click a single transformation.)
2. Click **Run Selected Transformations**. The portion of 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 following display shows a partial job that has been submitted.

**Display 7.1** Sample Submission of a Partial Job



Note that the **Run Selected Transformations** button is circled in the display. (The Sort transformation is also highlighted.) The following display shows the output from the partial submission.

**Display 7.2** Data from a Partial Submission


#	Name	Sex	Age	Height	Weight
1	Joyce	F	11	51.3	50.5
2	Jane	F	12	59.8	84.5
3	Louise	F	12	56.3	77
4	Alice	F	13	56.5	84
5	Barbara	F	13	65.3	98
6	Carol	F	14	62.8	102.5
7	Judy	F	14	64.3	90
8	Janet	F	15	62.5	112.5
9	Mary	F	15	66.5	112
10	Thomas	M	11	57.5	85
11	James	M	12	57.3	83
12	John	M	12	59	99.5
13	Robert	M	12	64.8	128
14	Jeffrey	M	13	62.5	84
15	Alfred	M	14	69	112.5
16	Henry	M	14	63.5	102.5
17	Ronald	M	15	67	133
18	William	M	15	66.5	112
19	Philip	M	16	72	150

Before the partial submission, the EMP\_SORT table was sorted by the Sex column. The partial submission added the Age column to the search. Note that the data is sorted first by sex and then by age.

### **Submit a Segment of a Job**

You can submit a segment of a job that either begins or ends at a selected transformation. To begin a job submission at a selected transformation, select the transformation and click **Run From Selected Transformation** on the Job Editor window toolbar. To end a job submission at a selected transformation, select the transformation and click **Run To Selected Transformation** on the toolbar.

### **Submit a Job One Step at a Time**

You can submit a job by running one step at a time. Click **Step** on the Job Editor window toolbar to move through the job on a step-by-step basis. You can click **Continue** on the toolbar to run the remainder of the job in a single submission.

### **Submit a Job to a Grid**

You can submit a job to a grid provided that the job is grid-enabled and the default SAS Application Server is configured for grid computing. To grid-enable a job, click **Yes** in the drop-down menu in the **Enable parallel processing macros** field on the **Options** tab of the properties window for the job.

For additional information about server requirements, system administrators should see the grid chapter in the *SAS Intelligence Platform: Application Server Administration Guide*.

If a Grid Server Component is available, you can select the component in the **Server** drop-down menu on the Job Editor window toolbar. Then, click **Submit** in the toolbar to submit the job to the grid.

---

## Meeting Prerequisites for Collecting Job Statistics

In order to track performance statistics for a SAS Data Integration Studio job, the following prerequisites must be met:

- The logging facility must be enabled on the SAS Workspace Server that executes the job. The logging facility is enabled by default. For more information, administrators should see the logging chapters in the *SAS Intelligence Platform System Administration Guide*.
- The collect runtime statistics option must be enabled for the job. To collect runtime statistics for an existing job, open the job in the Job Editor window. Then, right-click the canvas and select **Collect Runtime Statistics**. To collect runtime statistics for all new jobs, select **Tools** ⇒ **Options** ⇒ **Job Editor**. Then, select the check boxes for **Collect Runtime Statistics** and **Collect Table Statistics**.

*Note:* The collect runtime statistics option is on by default for a job. Some servers have the ARM statistics enabled by default (for example, workspace, batch, and other servers), but other servers do not (for example, stored process server).

---

## Reviewing a Successful Job

### Problem

You have run a successful job and want to review data about the job. You also want to examine the job output.

### Solution

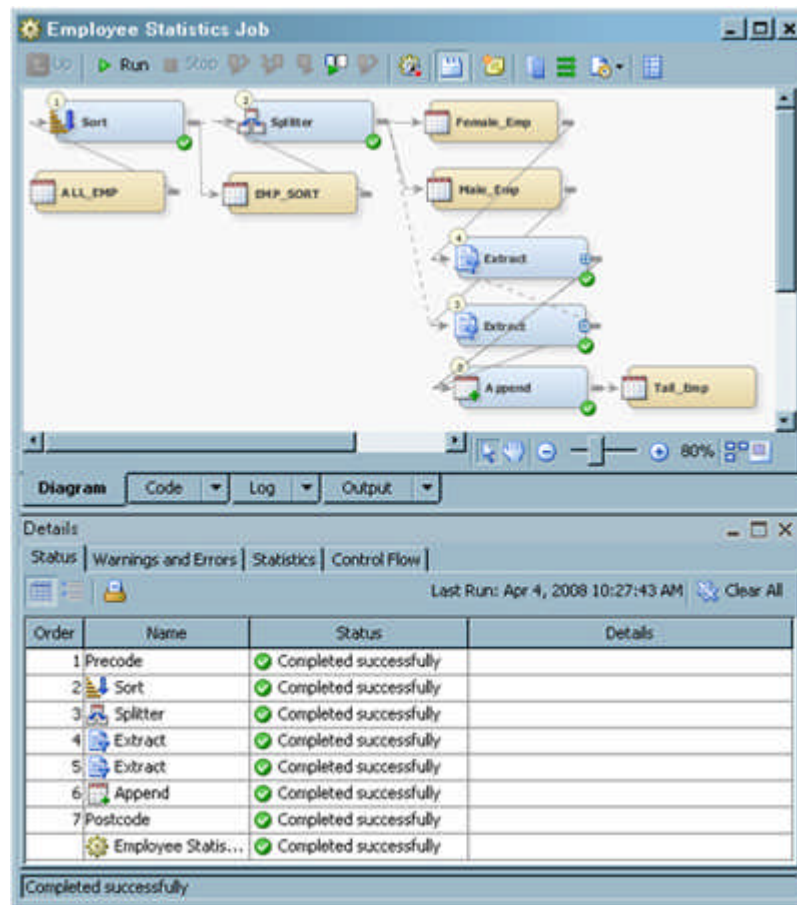
You can use the interactive tools that are provided with the Job Editor window. Perform the following tasks:

- [“Check the Status Tab” on page 145](#)
- [“Examine the Statistics Tab” on page 146](#)
- [“Examine the Control Flow Tab” on page 149](#)
- [“Review the Job Output” on page 149](#)

### Tasks

#### **Check the Status Tab**

Click **Status** in the Details section of the Job Editor window to display the status of each step in the job. If the Details section is not displayed, click **Details** in the **View** menu in the SAS Data Integration Studio menu bar. The following display shows a **Status** tab that confirms that all of the steps in a sample job were completed successfully.

**Display 7.3** Successfully Completed Sample Job

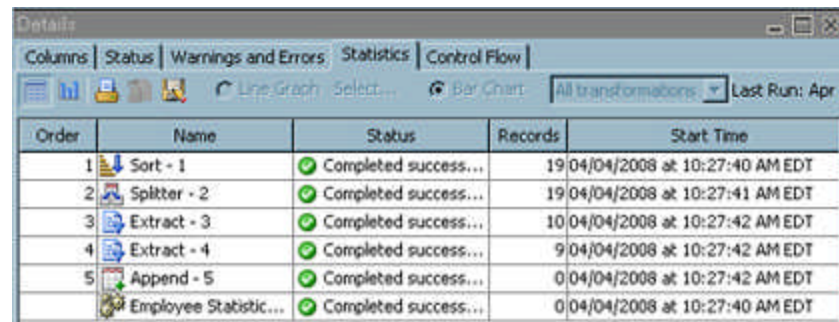
*Note:* The runtime status of each node in a job is also shown on the node on the **Diagram** tab. The following markers are placed on the jobs:

- a green check for a status of complete
- a yellow triangle for a warning
- red X for an error

In addition, you can review the basic properties of any object in the job. Click the object on the **Diagram**. Then, examine the Basic Properties pane for the object.

### **Examine the Statistics Tab**

Click **Statistics** in the Details section to display a tabular or graphic presentation of statistics about the progress of the job. Click the icon for the **Display table view for the statistics tab** on the Statistics toolbar to view a table of statistics. The following display shows the table for the sample job.

**Display 7.4** Sample Statistics Table


Order	Name	Status	Records	Start Time
1	Sort - 1	Completed success...	19	04/04/2008 at 10:27:40 AM EDT
2	Splitter - 2	Completed success...	19	04/04/2008 at 10:27:41 AM EDT
3	Extract - 3	Completed success...	10	04/04/2008 at 10:27:42 AM EDT
4	Extract - 4	Completed success...	9	04/04/2008 at 10:27:42 AM EDT
5	Append - 5	Completed success...	0	04/04/2008 at 10:27:42 AM EDT
	Employee Statistic...	Completed success...	0	04/04/2008 at 10:27:40 AM EDT

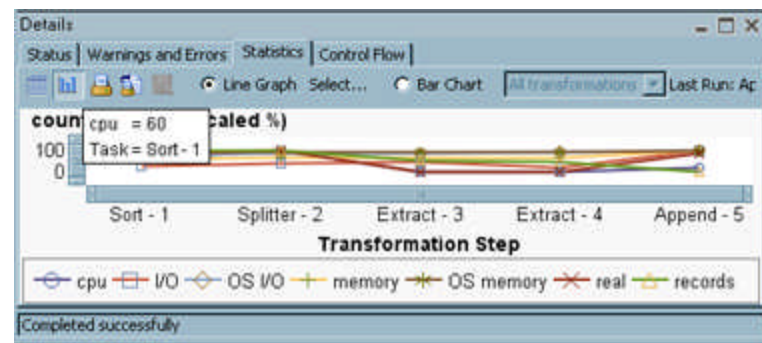
The statistics table includes the following columns:

- Order
- Name
- Status
- Records
- Start Time
- End Time
- Duration
- CPU Time
- Current Memory
- System Memory
- Current I/O
- System I/O
- Server
- Threads

You can click the icon for the **Display graph view for the statistics tab** on the Statistics toolbar to display a graphical chart. Select **Line Graph** to display a graph that charts one or more of the following values for the job:

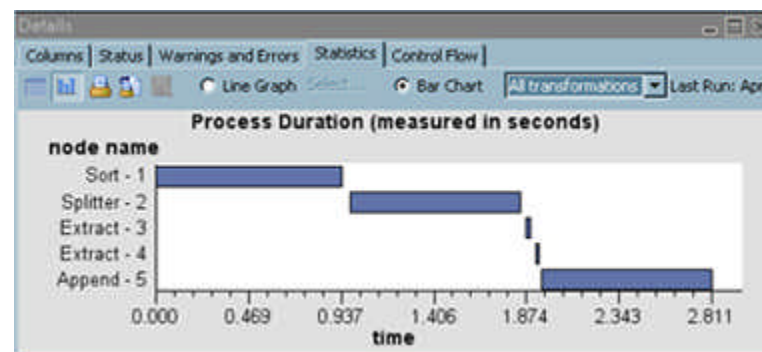
- CPU
- I/O
- OS I/O
- Memory
- OS Memory
- Real
- Records

Click **Select** to choose the values that are included in the graph. The following display shows a line graph of the sample job.

**Display 7.5** Sample Line Graph

Note that you can display a summary for a step in the job by positioning the cursor over its node.

Select **Bar Chart** to display a bar chart that illustrates the process duration of each transformation that is included in the job. Click **Select** to pick a single transformation or all transformations for inclusion in the graph. The following display shows a bar chart of the sample job.

**Display 7.6** Sample Bar Chart

You can display a detailed summary for a transformation by hovering the mouse over its bar.

If you don't see the output that you expect on the **Statistics** tab, then you can perform the following troubleshooting tasks:

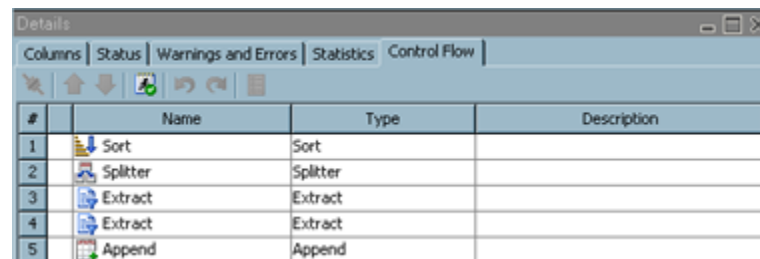
- When you execute jobs interactively and have runtime statistics enabled, output should be produced. If not, verify that the server is properly configured. See the "Use ARM to Display Runtime Statistics" section in the "Administering SAS Data Integration Studio" chapter of the *SAS Intelligence Platform: Desktop Application Administration Guide*.
- When runtime statistics and table counts are enabled but zero records are returned for the row count, verify that the table is not a view. A zero row count is returned for all views.
- Input and output counts are based on the input and output that are provided by the operating system. When a job has steps that are run on various operating systems, these numbers reflect the metrics that are returned by the operating system.



### Examine the Control Flow Tab

Click **Control Flow** in the Details section to access a table that consists of the transformations that are included in the job. These transformations are listed in the order in which they are run in the job. The following display shows the control flow table for the sample job.

**Display 7.7** Sample Control Table



#	Name	Type	Description
1	Sort	Sort	
2	Splitter	Splitter	
3	Extract	Extract	
4	Extract	Extract	
5	Append	Append	

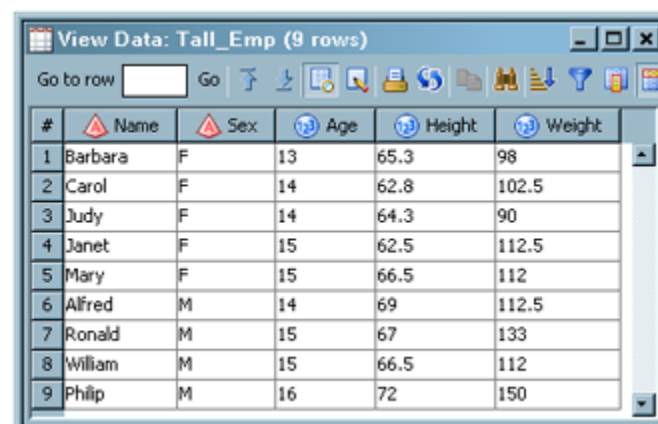
You can click **Validates the control flow** to make sure that the flow is valid. You can also drag a row to a higher or lower position in the table by clicking on the row number and moving the row either up or down. This action moves the transformation included in the row to a different position in the flow; it is run in an earlier or later position.

Control order is the order in which the nodes are run in a job. A warning in the control flow panel can be displayed when a step is ordered to run before the step that creates its data has run. For example, suppose there are two steps in a job in which Step 1 creates data that Step 2 uses, and Step 2 is ordered to run before Step 1. This arrangement forces Step 2 to run before its data is created. Step 2 is unlikely to run correctly because it doesn't have its data yet. If an out of order scenario is detected, then a warning icon is displayed to warn users that they might have steps out of order. However, they can still run the steps out of order if they choose.

### Review the Job Output

Right-click the target table of the job. Then, click **Open** in the pop-up window to see the output. The target table for the sample job is shown in the following display.

**Display 7.8** Sample View Data Window



#	Name	Sex	Age	Height	Weight
1	Barbara	F	13	65.3	98
2	Carol	F	14	62.8	102.5
3	Judy	F	14	64.3	90
4	Janet	F	15	62.5	112.5
5	Mary	F	15	66.5	112
6	Alfred	M	14	69	112.5
7	Ronald	M	15	67	133
8	William	M	15	66.5	112
9	Philip	M	16	72	150

You can also review basic details about the job in the Runtime Manager at the bottom of the SAS Data Integration Studio window. If the Runtime Manager is not displayed, click **Runtime Manager** in the **View** menu in the SAS Data Integration Studio menu bar. The Runtime Manager is shown in the following display.

**Display 7.9** Sample Runtime Manager

Runtime Manager		Actions History			
#	Job	Status	Start Time	End Time	Application Server Used
1	Employee Statistics ...	Complete	Apr 4, 2008 10:27:40 AM	Apr 4, 2008 10:27:43 AM	SASApp

## Diagnosing and Correcting an Unsuccessful Job

### Problem

You have run a job that was not successfully completed. You need to diagnose the problems with the job and correct them.

### Solution

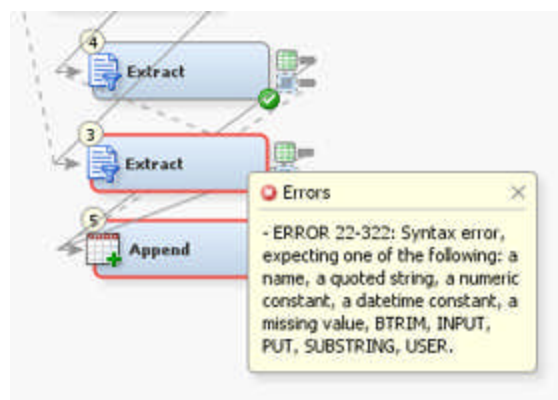
You can use the interactive tools that are provided with the Job Editor window. Perform the following tasks:

- “Examine the Diagram Tab” on page 150
- “Check the Status Tab” on page 151
- “Read the Warnings and Errors Tab” on page 151
- “Examine the Problem in the Log Tab” on page 152
- “Fix the Problem” on page 153
- “Run the Job and Check the Results” on page 153

### Tasks

#### Examine the Diagram Tab

You can easily see the transformations on the **Diagram** tab that generated error messages when the job was run. The transformations with errors are outlined in red and marked with a red dot in the bottom right corner. You can also click a red dot to see the error message in a sticky note window, as shown in the following display.

**Display 7.10** Transformation Error in a Sample Job

*Note:* When there are many warning or error messages, only the first few messages are shown in the sticky note due to performance reasons. You can set a limit on the number of messages at the following location: **Tools** ⇒ **Options** ⇒ **Job Editor** ⇒ **Maximum number of warnings and errors to display per step**.

### Check the Status Tab

Click **Status** in the Details section of the **Job Editor** window to display the status of each step in the job. If the Details section is not displayed, click **Details** in the **View** menu in the SAS Data Integration Studio menu bar. The following display shows a **Status** tab that shows that two of the steps in a sample job that resulted in errors.

**Display 7.11** Unsuccessful Sample Job

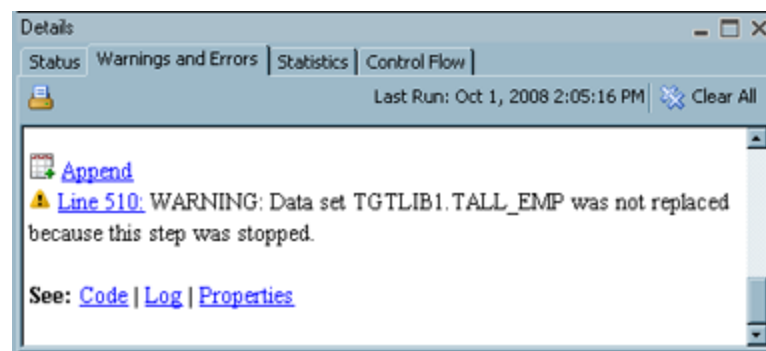
The screenshot shows the 'Employee Statistics Job' workflow in the Job Editor. The workflow consists of the following steps: 1. Precode, 2. Sort, 3. Splitter, 4. Extract (Female\_Emp), 5. Extract (Male\_Emp), 6. Append, and 7. Postcode. The 'Status' tab in the Details section shows the following results:

Order	Name	Status	Details
1	Precode	Completed successfully	
2	Sort	Completed successfully	
3	Splitter	Completed successfully	
4	Extract	Completed successfully	
5	Extract	Error	
6	Append	Error	
7	Postcode	Completed successfully	

At the bottom of the Status tab, it indicates '1 Warning, 2 Errors'.

### Read the Warnings and Errors Tab

Double-click on an error in the Status column of the **Status** tab to display the error in the **Warnings and Errors** tab.

**Display 7.12** Sample Warning and Errors Tab

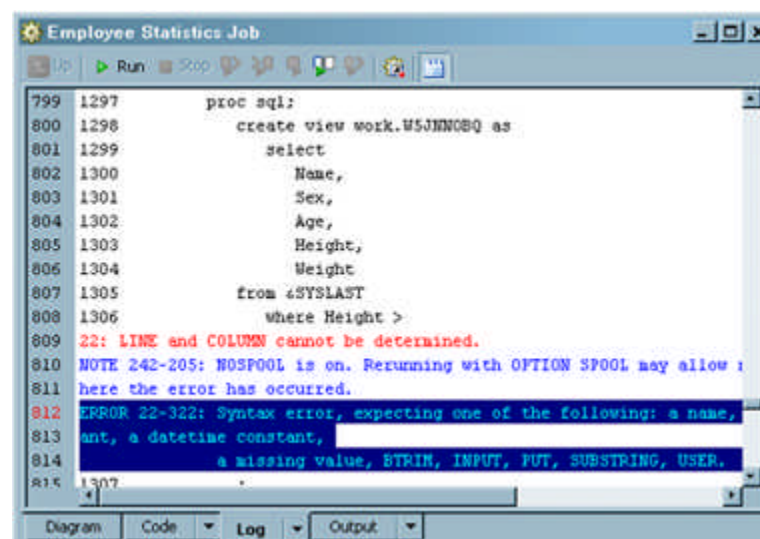
The following links are available on the **Warnings and Errors** tab to help you diagnose and correct the problem with the job:

- **The Transformation Name:** displays the transformation that is highlighted on the **Diagram** tab
- **Code:** displays the code for the transformation that is highlighted on the **Code** tab
- **Log:** displays the error on the **Log** tab
- **Properties:** displays the properties window for the transformation

### **Examine the Problem in the Log Tab**

Click **Log** on the **Warnings and Errors** tab to display the error on the **Log** tab. When you submit a job for execution, the SAS log is now updated at the end of each DATA step or procedure in the job. Therefore, you can use the SAS log to monitor the progress of each step in a job as it executes.

The following display shows the error in highlighted text. The log is scrolled to show both the error and the relevant lines in the code.

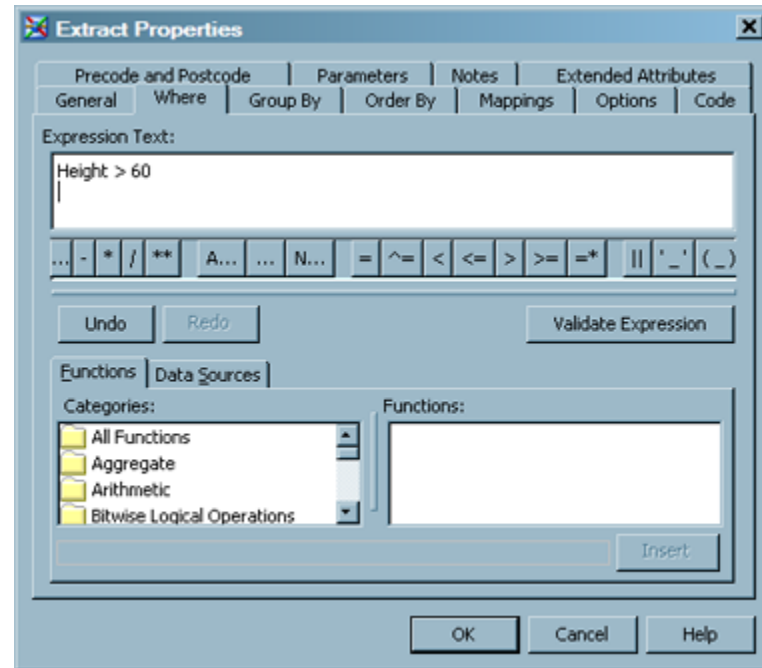
**Display 7.13** Sample Log Tab

The error corresponds to the code, which is missing a value for **where Height >**.

### Fix the Problem

Click **Properties** on the **Warnings and Errors** tab to display the properties tab for the appropriate transformation in the sample job. Then, click the appropriate tab and correct the error, as shown in the following display.

**Display 7.14** Sample Where Properties Tab



You can fix the sample job by correcting the text in the **Expression Text** field and saving the values in the properties window. After the correction, the expression text reads **Height > 60**.

### Run the Job and Check the Results

You can verify that the job is corrected. First, run the job and right-click the target table. Then, click **Open** in the pop-up menu to see the output. The target table for the sample job is shown in the following display.

**Display 7.15** Sample View Data Window

The screenshot shows the 'View Data: Tall\_Emp (9 rows)' window. It displays a table with 9 rows of employee data. The columns are: #, Name, Sex, Age, Height, and Weight. The data is as follows:

#	Name	Sex	Age	Height	Weight
1	Barbara	F	13	65.3	98
2	Carol	F	14	62.8	102.5
3	Judy	F	14	64.3	90
4	Janet	F	15	62.5	112.5
5	Mary	F	15	66.5	112
6	Alfred	M	14	69	112.5
7	Ronald	M	15	67	133
8	William	M	15	66.5	112
9	Philip	M	16	72	150

---

## 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. Mapping is the ability to create a relationship between a source and target column. The following mapping types are supported:

1-to-1

no expression is needed to create the column in the target from the source.

derived

an expression is required to create the column in the target based on the source.

### Solution

You create or maintain column mappings in the **Mappings** tab. The **Mappings** tab is available in the following places in a job:

- the Details section in the Job Editor window (when a transformation node is selected in the **Diagram** tab of the Job Editor window).
- the properties window for a transformation when the transformation has been added to the **Diagram** tab in the Job Editor window. The **Mappings** tab is not displayed in the properties window for a transformation in a tree or a folder.

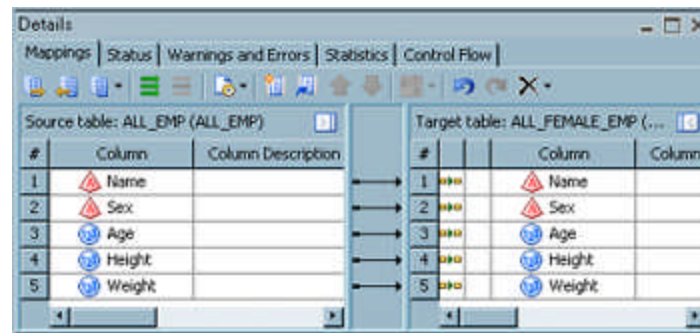
Perform the following tasks:

- [“Create Automatic Column Mappings” on page 154](#)
- [“Create One-to-One Column Mappings” on page 156](#)
- [“Create Derived Column Mappings” on page 156](#)
- [“Delete Column Mappings” on page 157](#)
- [“Use the Options for Mappings” on page 157](#)

### 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 **Mappings** tab. A **Mappings** tab from a sample job is shown in the following display.

**Display 7.16** Automatic Column Mappings

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:

- connecting a source and a target to the transformation on the **Diagram** tab
- clicking **Propagate** in the toolbar or in the pop-up menu in the Job Editor window
- clicking **Propagate** on the **Mappings** tab toolbar and selecting a propagation option
- clicking **Map all columns** on the **Mappings** tab toolbar

*Note:* When a transformation that is included in a job has multiple source or target tables, a drop-down menu is added to the top of the field. This menu enables you to select each individual table or all of the tables at once.

SAS Data Integration Studio might not be able to automatically create all column mappings that you need in a transformation. It 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 clicking **Propagate** or **Map all columns**.

You can use the following methods to disable automatic mapping:

- disable automatic mapping globally for new SAS Data Integration Studio jobs. Select or deselect **Automatically map columns** on the **Job Editor** tab in the Options window. To access the Options window, click **Options** in the **Tools** menu on the SAS Data Integration Studio menu bar.
- disable automatic mapping for the job. Deselect **Automatically Map Job** on the drop-down menu that is displayed when you click **Settings** on the toolbar at the top of the **Job Editor** window.
- disable automatic mapping for the transformation in a job. Deselect **Include Transformation in Mapping** on the drop-down menu that is displayed when you click **Settings** on the toolbar at the top of the **Mappings** tab.

*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 **Mappings** tab.
2. Click the column in the source table.
3. Hold down the CTRL key and click the column in the target table.
4. Click **Map selected columns** on the **Mappings** tab toolbar.

You can also create a mapping in the **Mappings** 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 can 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 that are illustrated in the following table to create different types of derived column mappings. All of the techniques are used on the **Mappings** tab in the properties window for the transformation.

**Table 7.1** Derived Column Techniques

Technique	Description
Directly enter an expression into an <b>Expression</b> field	<p>You can create any type of expression by entering the expression directly into an <b>Expression</b> 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:</p> <ol style="list-style-type: none"> <li>1. Double-click in the field in which you want to enter the expression. A cursor displays in the field. (The button disappears.)</li> <li>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 &amp;SYSDATE.</li> </ol>
Create expressions that use no source columns	<p>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:</p> <ol style="list-style-type: none"> <li>1. Right-click in an Expression column. Then, click Advanced in the pop-up menu to access the Expression window.</li> <li>2. Use the Expression Builder to create an expression. Then, click OK to save the expression, close the Expression window, and display the expression in the selected column in the target table.</li> </ol>



Technique	Description
Create expressions that use a single source column	<p>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 is possible if the discount is a constant, such as 6 percent. That is, you might want to define an expression as <b>Price * .94</b>. You could perform the following steps:</p> <ol style="list-style-type: none"> <li>1. Select the Price source column and the DiscountedPrice target column.</li> <li>2. Right-click either selected variable, and select Expression from the pop-up menu. Then, select Advanced to access the Expression window.</li> <li>3. Use the Expression Builder to create an expression. Then, click OK to save the expression, close the Expression window, and display the expression in the selected column in the target table.</li> </ol>
Create expressions that use two or more source columns	<p>You can create a derived mapping that uses two or more source columns. Perform the following steps:</p> <ol style="list-style-type: none"> <li>1. Select the source columns and target column to be used in the mapping. For example, you can use the values of the Price and Discount columns in the source in an expression. Then, the result can be written to the DiscountedPrice column in the target.</li> <li>2. Review the warning that displays because two source columns are mapped to a single target column.</li> <li>3. Right-click either selected variable, and click Expression from the pop-up menu. Then, select Advanced from the submenu to access the Expression window.</li> <li>4. Create the expression, which is <math>Price - (Price * (Discount / 100))</math> in this example. Then, click OK to save the expression, close the Expression window, and display the expression in the selected column in the target table.</li> </ol>

### Delete Column Mappings

You can delete a column mapping in the **Mappings** tab by using one of the following methods:

- 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 Mappings** in the pop-up menu.

*Note:* You must disable automatic mapping for a transformation in order to delete mappings that are otherwise automatically created.

### Use the Options for Mappings

You can use the toolbar or the pop-up menu in the **Mapping** tab of the properties window to control the behavior of the tab. To access the Help for the **Mapping** tab, click on the **Help** button at the top of the SAS Data Integration Studio window. Under the folder for Windows and Other Components, select the **Popup Menus** icon. Click on the Pop-Up Menu Options for Mapping link.

---

## Managing the Scope of Column Changes in Jobs

### Problem

You have added columns and you need to determine the scope of these additions. Select one of the following scenarios:

- No propagation: Adding column changes to the output of a single transformation in a job
- Automatic propagation: Automatically adding column changes to tables in a specified direction
- Manual propagation: Manually controlling the addition of column changes in specified paths and directions

Note that you can propagate column changes only in the context of a job. If you add column changes in the properties window for a table from a tree or a folder, the propagate and mapping options that you see on the **Mappings** tab in a job are not available. In that case, you must remember to map and propagate the column changes when you later use the altered table in a job. Therefore, it is generally more efficient to make and propagate your columns directly in the jobs where you need them.

### Solution

You can use an appropriate propagation control in a SAS Data Integration Studio job to enable or disable automatic propagation or to exercise manual control over propagation functions. Perform the following tasks:

- [“Managing Automatic Propagation” on page 158](#)
- [“Managing Manual Propagation” on page 159](#)

### Tasks

#### **Managing Automatic Propagation**

Automatic propagation sends column changes to tables when process flows are created. If you disable automatic propagation and refrain from using manual propagation, you can propagate column changes on the **Mappings** tab for a transformation that are restricted to the target tables for that transformation. Automatic propagation controls are explained in the following table.

**Table 7.2** Automatic Propagation Controls

Level	Control	Set Propagation Direction
Global	<b>Automatically propagate columns</b> in the <b>Automatic Settings</b> group box on the <b>Job Editor</b> tab in the Options window. (Click <b>Options</b> in the <b>Tools</b> menu to display the window.) This option controls automatic propagation of column changes in all new jobs.	Select one of the following directions in the Propagation Direction group box: <ul style="list-style-type: none"> <li>From beginning to end</li> <li>From end to beginning</li> </ul>
Job	<b>Automatically Propagate Job</b> in the drop-down menu that displays when you click <b>Settings</b> in the toolbar on the <b>Diagram</b> tab in the Job Editor window. This option controls automatic propagation of column changes in the currently opened job.	Select one of the following directions in the drop-down menu: <ul style="list-style-type: none"> <li>From Beginning to End</li> <li>From End to Beginning</li> </ul>
Process flow	<b>Propagate Columns</b> in the pop-up menu on the <b>Diagram</b> tab in the Job Editor window. This option controls automatic propagation of column changes in the process flow in a currently opened job.	Select one of the following directions in the pop-up menu: <ul style="list-style-type: none"> <li>To Beginning</li> <li>To End</li> </ul>
Transformation	<b>Include Transformation in Propagation</b> in the drop-down menu that displays when you click <b>Settings</b> in the toolbar on the <b>Mappings</b> tab. This option controls automatic propagation of column changes in the selected transformation.	Not applicable
Transformation	<b>Include Selected Columns in Propagation</b> in the drop-down menu that displays when you click <b>Settings</b> in the toolbar on the <b>Mappings</b> tab to propagate changes to columns that you select in the source or target tables for the selected transformation.	Not applicable

The **Mappings** tab is available in the following locations:

- the Details section in the Job Editor window
- the properties windows for any transformation that is included on the **Diagram** tab of the Job Editor window

The **Mappings** tab performs the same functions and contains the same items in both locations.

### **Managing Manual Propagation**

Add, delete, or update the columns in your job. Manual propagation controls are explained in the following table.

**Table 7.3** Manual Propagation Options

Level	Control	Function	Direction
Job	<b>Propagate Job</b> in the toolbar in the <b>Diagram</b> tab in the Job Editor window	Propagates column changes in the job.	Uses the direction set with Settings on the Job Editor toolbar.
Process flow	<b>Propagate Columns</b> in the pop-up menu in the <b>Diagram</b> tab in the Job Editor window	Propagates column changes in the process flow in a specified direction.	Use the following directions: <ul style="list-style-type: none"> <li>To Beginning</li> <li>To End</li> </ul>
Transformation	<b>Propagate from sources to targets</b> in the toolbar in the <b>Mappings</b> tab	Propagates column changes in the process flow from source tables to target tables.	From source tables to target tables.
Transformation	<b>Propagate from targets to sources</b> in the toolbar in the <b>Mappings</b> tab	Propagates column changes in the process flow from target tables to source tables.	From target tables to source tables.
Transformation	<b>Propagate</b> in pop-up menus in the <b>Source table</b> field and the <b>Target table</b> field	Specifies a path and a direction for propagating column changes. See the table that follows for details.	
Transformation	<b>Propagate columns</b> in the toolbar on the <b>Mappings</b> tab	Specifies a path and a direction for propagating column changes. See the table that follows for details.	

The following table specifies the available path and direction options for the **Propagate** field and **Propagate columns** field on the **Mappings** tab for a transformation.

**Table 7.4** Propagation Path Options

Path	Direction
For the <b>Propagate</b> option in pop-up menus in the <b>Source table</b> field and the <b>Target table</b> field	
To Targets	<ul style="list-style-type: none"> <li>From Sources</li> <li>From Beginning</li> <li>From End</li> </ul>
From Targets	<ul style="list-style-type: none"> <li>To Sources</li> <li>To Beginning</li> <li>To End</li> </ul>

Path	Direction
Selected Target Columns	<ul style="list-style-type: none"> <li>• To Sources</li> <li>• To Beginning</li> <li>• To End</li> </ul>
Update Selected Target Columns	<ul style="list-style-type: none"> <li>• To Sources</li> <li>• To Beginning</li> <li>• To End</li> </ul>
For the <b>Propagate columns</b> in the toolbar on the <b>Mappings</b> tab	
To Targets	<ul style="list-style-type: none"> <li>• From Sources</li> <li>• From Beginning</li> <li>• From End</li> </ul>
To Sources	<ul style="list-style-type: none"> <li>• From Targets</li> <li>• From Beginning</li> <li>• From End</li> </ul>
From Targets	<ul style="list-style-type: none"> <li>• To Sources</li> <li>• To Beginning</li> <li>• To End</li> </ul>
From Sources	<ul style="list-style-type: none"> <li>• To Targets</li> <li>• To Beginning</li> <li>• To End</li> </ul>
Selected Target Columns	<ul style="list-style-type: none"> <li>• To Sources</li> <li>• To Beginning</li> <li>• To End</li> </ul>
Selected Sources Columns	<ul style="list-style-type: none"> <li>• To Targets</li> <li>• To Beginning</li> <li>• To End</li> </ul>
Update Selected Target Columns	<ul style="list-style-type: none"> <li>• To Sources</li> <li>• To Beginning</li> <li>• To End</li> </ul>
Update Selected Sources Columns	<ul style="list-style-type: none"> <li>• To Targets</li> <li>• To Beginning</li> <li>• To End</li> </ul>

---

## Managing Connections in Job Editor Windows

### Problem

You need to manage the input and output connections for the objects in a SAS Data Integration Studio job. For example, you might need to switch an input table for a transformation with an output table.

### Solution

You can use the Connections window for an object on the **Diagram** tab in the Job Editor window to review or change the input and output connections for the object. You can access the Connections window for the following objects:

- a table
- a transformation
- a temporary output table

Perform the following tasks:

- [“Review the Connections for the Object” on page 162](#)
- [“Change the Inputs and Outputs for the Object” on page 163](#)

### Tasks

#### **Review the Connections for the Object**

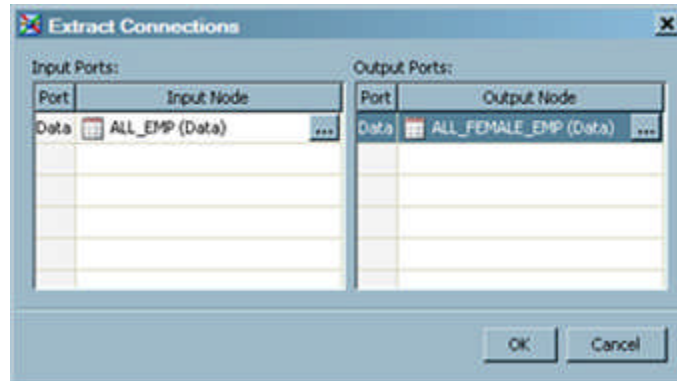
The Connections window displays the input and output nodes for any selected object in the Job Editor window. For example, you can display the Connections window for an object in the sample job shown in the following display.

**Display 7.17** Initial Process Flow



Perform the following steps to review the connections for an object in the job.

1. Right-click the object that you need to review. Then, click **Connections** in the pop-up menu to display the Connections window. The following display shows the Connections window for the Extract transformation in the sample job.

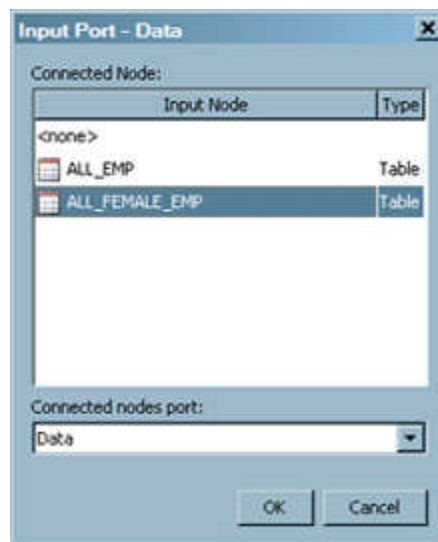
**Display 7.18** Connections Window

- Review the inputs and outputs for the object. Note that the ALL\_EMP table is listed as an input node in the **Input Ports** field. In addition, the ALL\_FEMALE\_EMP is listed as an output node in the **Output Ports** field. Both fields also include a **Selector** button. This button is displayed only when the node can be deleted or replaced with another object in the job.

### **Change the Inputs and Outputs for the Object**

The input and output selector windows enable you to change the connections in and out of the objects that are contained in the job. Perform the following steps to display and use a selector window.

- Click the **Selector** button to display the selector window for an input or output node. The following display shows the Input Selector window for the Extract transformation in the sample job.

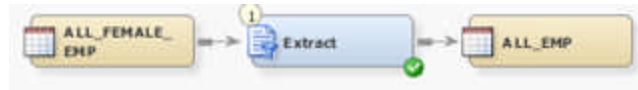
**Display 7.19** Input Selector Window

Note that the **Connected Node** field contains the input and the output tables for the job. The field also contains a **<none>** field, which you can use to remove the input table to the transformation entirely. The display shows the target table, ALL\_FEMALE\_EMP selected.

- Click **OK** to save the change to the input node for the object.
- Use selector windows to change any other objects that you need to update. Then, save the changes.

- Click **OK** in the Connections window to close the window and save the changes to the job. The following display shows the updated sample job after the source and target tables are dragged to their appropriate places on the **Diagram** tab.

**Display 7.20** Updated Process Flow



The source table and the target table have exchanged places.

---

## 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 the metadata for a transformation in the transformation's Code window. This window is available only when the transformation is included in a SAS Data Integration Studio job.

### Tasks

#### **View the Code in a Transformation**

Perform the following steps to access the code in a transformation that is included in a SAS Data Integration Studio job:

- Open an existing SAS Data Integration Studio job.
- Right-click the transformation in the Job Editor window that contains the code that you want to review. Then, click **Properties** in the pop-up menu to access the properties window for the transformation.
- Open the **Code** tab, and review the code for the transformation.
- Click **View Step Code** to access the View Step Code window. Review the code for the step in the job that includes the selected transformation.
- Close the View Step Code window and the properties window for the 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***

Perform the following steps to access the metadata for a transformation that is included in a SAS Data Integration Studio job:

1. Open an existing SAS Data Integration Studio job.
2. Right-click the transformation in the Job Editor 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.



## Chapter 8

# Managing the Status of Jobs and Transformations

---

<b>About Status Handling for Jobs and Transformations</b> . . . . .	<b>167</b>
<b>Default Conditions, Actions, and Conditional Action Sets</b> . . . . .	<b>168</b>
Overview . . . . .	168
Default Conditions . . . . .	168
Default Actions . . . . .	169
Conditional Action Sets . . . . .	171
<b>Prerequisites for Actions</b> . . . . .	<b>172</b>
<b>Perform Actions Based on the Status of a Job</b> . . . . .	<b>173</b>
Problem . . . . .	173
Solution . . . . .	174
Tasks . . . . .	174
<b>Perform Actions Based on the Status of a Transformation</b> . . . . .	<b>175</b>
Problem . . . . .	175
Solution . . . . .	175
Tasks . . . . .	175
<b>Macro Variables for Status Handling</b> . . . . .	<b>177</b>
Overview . . . . .	177
Example: Macro Variables for Status Handling in Generated Code . . . . .	177
Macro Variables for Status Handling in User-Written Code . . . . .	182

---

## About Status Handling for Jobs and Transformations

When you execute a SAS Data Integration Studio job, 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. These return codes can be used to test for certain conditions, such as **Successful** or **Lookup Failed**. Use the **Status Handling** tab in the property window for jobs and transformations to specify an action that should be performed when a certain condition is met, such as **Send Email** or **Send Event**. In this way, you can specify actions based on the status of a job or transformation when it is executed.

For example, if a lookup fails in the process flow for a job, the job can be terminated, and a status message can be sent to a person, to a file, or to an event broker that passes the status message to another application. You can also use status handling to capture job statistics, such as the number of records before and after an append of the last table loaded in the job.

To capture statistics about a job, select the desired condition to be tested for the job, such as **Successful**, then associate that condition with the **Send Job Status** action.

## Default Conditions, Actions, and Conditional Action Sets

### Overview

SAS Data Integration Studio provides a number of default conditions, actions, and condition action sets. These are displayed in the Inventory tree and the Folders tree. Typically, however, you do not interact with these objects in the tree view. Instead, you use the **Status Handling** tab in the property windows of jobs and transformations.

*Note:* If you want to add user-defined condition templates, action templates, or condition action set templates, contact your SAS representative.

### Default Conditions

All of the default conditions are listed in the following table and in the **Condition** folder in the Inventory tree. Only those conditions that are valid for a job or for a specific kind of transformation are displayed on the **Status Handling** tab.

**Table 8.1** Default Conditions

Condition	Description
Data Exception	An exception occurred as the Data Validation transformation processed data.
Data Modified	The transformation modified data.
Errors in Process	There was an error in a process.
Errors	This checks for return code > 4.
Lookup Failed	The lookup value was not found.
Lookup Table Missing	The lookup table is missing.
No Lookup Rows	There are no rows in the lookup table.
Send Job Status	The job status table is created.
Successful	This checks for return code=0.
Successful RC=1, RC=2, and RC=3	This condition is not used.
Table Created	A table is created in physical storage.
Table Does Not Exist	Table does not exist in physical storage.

Condition	Description
Table Dropped	The table is deleted.
Table Not Match Meta	This identifies when the table does not match the metadata.
Table Truncated	The table is truncated.
Warnings	This checks for return code > 3.

## Default Actions

You can specify an action that should be performed when a certain condition is met. When you select a condition on the **Status Handling** tab, only those actions that are valid for that condition are available to be selected. The Input column in the following table describes the values that are required by some actions.

**Table 8.2** Default Actions

Action	Description	Input
Abort	Terminates the job or transformation.	None.
Abort After Looping	Completes all of the processes in the loop and then terminates the job.	None.
Abort All Processes	Terminates all of the currently executing and remaining processes.	None.
Abort Remaining	Terminates all of the remaining processes after the current process executes.	None.
Add Row to Error Table	Adds a row to an error table for a Lookup transformation.	None.
Add Row to Exception Table	Adds a row to an exception table, as specified by the transformation.	None.
Custom	Calls SAS code to provide user-defined status handling for a job or transformation. Examples include SAS code added to the <b>Precode and Postcode</b> tab in a job or transformation, or a macro in a SAS Autocall library.	In the <b>Custom Code</b> field, enter a call to the user-defined code. One example is the following call to a macro in a SAS Autocall library: <b>%sendcustom;</b>
Do Not Create Report	Prevents the creation of an exception report.	None.
Email Report	Sends an exception report to the specified e-mail address.	E-mail address.

Action	Description	Input
Save Report	Saves the exception report to the specified location.	Location for the exception report.
Save Table	Saves status messages to a table. Consecutive messages are appended to the table with a timestamp.	Table name in the LIBREF.DATASET SAS format. The libref must be assigned before the job or transformation executes.
Send Email	Sends an e-mail message that you specify.	One or more recipient e-mail addresses and a message in the options window. 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 <b>Message</b> field that includes white space must be enclosed by single quotation marks so that the mail is processed correctly.
Send Entry to Data Set	Saves status messages to a SAS data set. Consecutive messages are appended to the data set with a timestamp.	Data set name in the LIBREF.DATASET SAS format. The libref must be assigned before the job or transformation executes.
Send Entry to File	Sends text to the specified filename.	Physical path to a file; text of the message.
Send Event	If an event broker is configured, this action sends a status message to the 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	Updates the job status table with a record when the current job completes.	Data set name in the LIBREF.DATASET SAS format. The libref must be assigned before the job or transformation executes.
Set Target Column Value	Sets the target column to the specified value; accessible from the <b>Lookups</b> tab of the Lookup transformation property window.	SAS expression.

Action	Description	Input
Set Target Column Value to Missing	Sets the target column value to missing; accessible from the <b>Lookups</b> tab of the Lookup transformation property window.	None.
Skip the Record	Skips a record that has an error.	None.

## Conditional Action Sets

All of the default action sets are listed in the following table and in the **Conditional Action Sets** folder in the Inventory tree. Typically you do not interact with these sets. They provide status handling for the standard SAS Data Integration Studio transformations.

**Table 8.3** Default Conditional Action Sets

Conditional Action Sets	Description
Data Exception	Condition: Data Exception Actions: None, Send Email, Send Entry to Dataset, Send Entry to File, Send Event, Do not create report, Email Report, Save Report, Save Table
Send Job Status	Condition: Send Job Status Actions: None, Send Job Status
Set Data Modified	Condition: Data Modified Actions: None, Custom, Send Email, Send Entry to Dataset, Send Entry to File, Send Event
Set Error in Process	Condition: Error in Process Actions: None, Custom, Send Email, Send Entry to Dataset, Send Entry to File, Abort All Processes, Abort Remaining, Abort After Looping, Send Event
Set Errors	Condition: Errors Actions: None, Custom, Send Email, Send Entry to Dataset, Send Entry to File, Abort, Send Event
Set Lookup Not Found	Condition: Lookup Failed Actions: None, Abort, Add Row to Error Table, Add Row to Exception Table, Set Target Column Value, Set Target Column Value to Missing, Skip the Record
Set Lookup Table Missing	Condition: Lookup Table Missing Actions: None, Abort, Add Row to Error Table, Add Row to Exception Table, Set Target Column Value, Set Target Column Value to Missing, Skip the Record

Conditional Action Sets	Description
Set Lookup Table Missing Records	Condition: No Lookup Rows Actions: None, Abort, Add Row to Error Table, Add Row to Exception Table, Set Target Column Value, Set Target Column Value to Missing, Skip the Record
Set Successful	Condition: Successful Actions: None, Custom, Send Email, Send Entry to Dataset
Set Successful return code =1	Not used
Set Successful return code =2	Not used
Set Successful return code =3	Not used
Set Table Created	Condition: Table Created Actions: None, Custom, Send Email, Send Entry to Dataset
Set Table Different	Condition: Table Different Actions: None, Custom, Send Email, Send Entry to Dataset, Send Entry to File, Send Event
Set Table Does Not Exist	Condition: Table Does Not Exist Actions: None, Custom, Send Email, Send Entry to Dataset, Send Entry to File, Send Event
Set Table Dropped	Condition: Table Dropped Actions: None, Custom, Send Email, Send Entry to Dataset, Send Entry to File, Send Event
Set Table Truncated	Condition: Table Truncated Actions: None, Custom, Send Email, Send Entry to Dataset, Send Entry to File, Send Event
Set Warnings	Condition: Warnings Actions: None, Custom, Send Email, Send Entry to Dataset, Send Entry to File, Send Event

---

## Prerequisites for Actions

Some actions that can be selected on the **Status Handling** tab require server setup, as described in the following table.



**Table 8.4** Prerequisites for Status Handling Actions

Action	Description
Any action that sends e-mail.	E-mail must be enabled for the SAS Workspace Server that executes the job that includes the action. For more information, administrators should see the section called "Add or Modify E-Mail Settings for SAS Application Servers" in the <i>SAS Intelligence Platform: Application Server Administration Guide</i> .
Send Event	SAS Foundation Services must be installed, and the Event Broker Service must be properly configured for the software that receives the events. For more information, see the documentation for SAS Foundation Services and for the software that receives the events.
Custom	<p>The Custom action enables you to call SAS code to provide user-defined status handling for a job or transformation. Examples include SAS code that is added to the <b>Precode and Postcode</b> tab in a job or transformation, or a macro in a SAS Autocall library. The SAS code must have valid SAS syntax based on the location it is being called from.</p> <p>If you call a macro in a SAS Autocall library, the SAS Application Server that executes the job must be able to access the relevant Autocall library. For details about making Autocall macro libraries available to SAS Data Integration Studio, see the "Administering SAS Data Integration Studio" chapter in the <i>SAS Intelligence Platform: Desktop Application Administration Guide</i>.</p>
Any action that requires a libref	<p>The libref must be assigned before the job or transformation executes. To assign a library within SAS Data Integration Studio, you can select the <b>Pre and Post Process</b> tab in the properties window for the job or transformation and then specify a SAS LIBNAME statement as a preprocess.</p> <p>To assign a library outside of SAS Data Integration Studio, you can pre-assign the library to the SAS Application Server that is used to execute the job. Some tasks that are associated with pre-assigning a SAS library must be done outside of SAS Data Integration Studio or SAS Management Console. For details, see the "Assigning Libraries" chapter in <i>SAS Intelligence Platform: Data Administration Guide</i>.</p>

*Note:* If an action requires you to specify a physical path, then use relative paths for portability.

---

## Perform Actions Based on the Status of a Job

### Problem

When a job is executed, you want certain actions to be performed automatically based on the status of the job.

## Solution

You can use the **Status Handling** tab in the properties window for a job to specify one or more pairs of conditions and actions. These conditions and actions apply to the job as a whole.

Perform the following tasks:

- “Specify Conditions and Actions for the Job” on page 174
- “Run the Job and Verify the Status Handling Output” on page 174

Some actions require server setup, as described in “Prerequisites for Actions” on page 172.

## Tasks

### **Specify Conditions and Actions for the Job**

Perform the following steps to specify actions to be performed automatically based on the status of a job.

1. Right-click the job in a tree view and select **Properties** from the menu.
2. Click the **Status Handling** tab.
3. Click **New**. A default condition and action are displayed in the first row of the table.
4. To replace the default condition, use the selection arrow to select another condition, such as **Error**.
5. To replace the default action, use the selection arrow to select another action, such as **Send Email**. If the action requires information from you, the Action Options window appears.
6. Use the Action Options window to specify any values that are required by the action. For example, a **Send Email** action requires an e-mail address.
7. Select more conditions and actions, as desired.
8. Click **OK** to close the properties window.

### **Run the Job and Verify the Status Handling Output**

Perform the following steps to run the job and verify the status handling output.

1. Right-click the job in a tree view and select **Open** from the menu. The job opens in the Job Editor.
2. Click **Run**.
3. If any of the conditions that you specified are met, then the actions that you specified should be performed.

## Perform Actions Based on the Status of a Transformation

### Problem

When a job is executed, you want certain actions to be performed automatically based on the status of a transformation in the job.

### Solution

If the transformation has its own **Status Handling** tab, you can use this tab to specify one or more pairs of conditions and actions for the transformation. If the transformation does not have its own **Status Handling** tab, you can insert a Return Code Check transformation into the process flow, after the transformation that you want to monitor. A Return Code Check transformation can specify conditions and actions for the preceding transformation in a process flow.

Accordingly, use one of the following methods:

- [“Use the Status Handling Tab for the Transformation You Want to Monitor” on page 175](#)
- [“Add a Return Code Check Transformation After the Transformation You Want to Monitor” on page 176](#)

Then verify the job as described in [“Run the Job and Verify the Status Handling Output” on page 176](#). Some actions require server setup, as described in [“Prerequisites for Actions” on page 172](#).

### Tasks

#### ***Use the Status Handling Tab for the Transformation You Want to Monitor***

Perform the following steps when a transformation has its own **Status Handling** tab, and you want to specify actions to be performed automatically based on the status of the transformation.

1. Right-click the appropriate job in a tree view and select **Open** from the menu. The job opens in the Job Editor.
2. Right-click the desired transformation in the process flow and select **Properties** from the menu
3. Click the **Status Handling** tab.
4. Click **New**. A default condition and action are displayed in the first row of the table.
5. Some transformations check for only one status condition. Others might have several conditions to choose from. To replace the default condition, use the selection arrow to select another condition, such as **Error**.
6. To replace the default action, use the selection arrow to select another action, such as **Send Entry to File**. If the action requires information from you, the Action Options window appears.

7. Use the Action Options window to specify any values that are required by the action. For example, a **Send Entry to File** action requires a physical path to a file.
8. Select more conditions and actions, as desired.
9. Click **OK** to close the properties window.

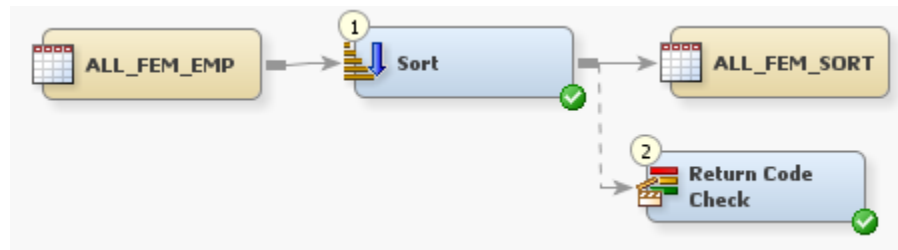
You are now ready to run the job and verify the status handling output.

### **Add a Return Code Check Transformation After the Transformation You Want to Monitor**

Perform the following steps when a transformation does not have its own **Status Handling** tab, and you want to specify actions to be performed automatically based on the status of the transformation.

1. Right-click the appropriate job in a tree view and select **Open** from the menu. The job opens in the Job Editor.
2. Open the **Control** folder in the Transformations tree. Right-click the Return Code Check transformation, and then select **Add to Diagram**. The Return Code Check transformation is added to the end of the process flow of the job. The next display shows an example process flow for a job with a Return Code Check transformation.

**Display 8.1** Process Flow with a Return Code Check Transformation



3. Verify that Return Code Check transformation will be executed immediately after the transformation that you want to monitor. For example, in the preceding display, the Return Code Check transformation is executed immediately after the Sort transformation. Any actions and conditions that are specified in the Return Code Check transformation are applied to the Sort transformation.

If you need to change the execution order of the transformations in a process flow, select **View** ⇒ **Details** from the menu bar on the desktop. On the Details pane, click **Control Flow** tab. Use that tab to change the execution order of the transformations.

4. To specify actions and conditions, right-click the Return Code Check transformation in the process flow and select **Properties** from the menu.
5. Click the **Status Handling** tab.
6. Use the **Status Handling** tab to specify conditions and actions, as described in [“Use the Status Handling Tab for the Transformation You Want to Monitor” on page 175](#). These conditions and actions are checked for the preceding transformation in the process flow.
7. Click **OK** to close the properties window.

You are now ready to run the job and verify the status handling output.

### **Run the Job and Verify the Status Handling Output**

Perform the following tasks to run the job and verify the status handling output.

1. Right-click the appropriate job in a tree view and select **Open** from the menu. The job opens in the Job Editor.
2. Click **Run**.
3. If any of the conditions that you specified are met, the actions that you specified should be performed.

---

## Macro Variables for Status Handling

### Overview

The following topics examine the use of macro variables in status handling:

- [“Example: Macro Variables for Status Handling in Generated Code” on page 177](#)
- [“Macro Variables for Status Handling in User-Written Code” on page 182](#)

When SAS Data Integration Studio generates the code for a job, the code includes the following macro and macro variables:

- **RCSET**: This macro sets the values of the TRANS\_RC and JOB\_RC variables. Accepts numeric values or autocall macros as parameters. For example, you can pass a numeric value of 9999 to RCSET, using the following syntax:

```
%RCSET(9999) ;
```

You can also pass one of the following autocall macros to RCSET:

- **&syserr** — used to set TRANS\_RC and JOB\_RC for SAS procedures and the SAS DATA STEP.
- **&syslibrc** — used to set TRANS\_RC and JOB\_RC for SAS LIBNAME statements.
- **&sqlrc** — used to set TRANS\_RC and JOB\_RC for the SQL procedure and pass-through statements.

The syntax is as follows:

```
%RCSET(&syslibrc) ;
```

- **TRANS\_RC**: This variable is cleared at the beginning of generated code for each transformation. The RCSET macro resets the TRANS\_RC variable after each library assignment statement and after the main generated code for the transformation. If the transformation has more than one processing step, then the TRANS\_RC macro is set to the highest value.
- **JOB\_RC**: This variable is set to 0 at the top of the job. It is not cleared as the code for the job is executed. At the end of the job, the RCSET macro sets the JOB\_RC variable to the highest return code value of the entire job.

### Example: Macro Variables for Status Handling in Generated Code

Suppose that you created a simple job in which a SAS table named ADVERSE is loaded into another SAS table named ADVERSE2. There is a one-to-one mapping of columns from ADVERSE to ADVERSE2. SAS Data Integration Studio generates the following code for this job. Note how the status handling macro and macro variables are used.

```

/*-----
* Name: Simple Load Job
* Description: Code generated for Server SASMain
* Generated: Tue Jun 29 13:29:09 EDT 2008
*-----*/
/* This is the setup required to capture the transformation return code */
%let JOB_RC=0;
%let TRANS_RC=0;
%global SQLRC;
%global SYSERR;

%macro RCSET(error);
%if (&error gt &TRANS_RC) %then
%let TRANS_RC=&error;
%if (&error gt &JOB_RC) %then
%let JOB_RC=&error;
%mend RCSET;

%let TRANS_RC=0;

options VALIDVARNAME=ANY;
/*
* Access the data for Test_lib
*/
LIBNAME testlib BASE "C:\sources\test";

%RCSET(&syslibrc);

%let SYSLAST=%nrquote(testlib."ADVERSE"n);

/*****
* Name: Loader
* Description: Codegen
* Generated: Tue Jun 29 13:29:09 EDT 2008
*****/
%let SYSOPT=;

%global DBXRC;
%global DWNUMIDX;
%global DBXLAST;
%let DBXRC=-1;
%let DWNUMIDX=-1;
%let DBXLAST=&SYSLAST;

/*-----
* Name: DBWALOAD
* Description: Define load data macro
* Generated: Tue Jun 29 13:29:09 EDT 2008
*-----*/
%macro dbwaload;

/* Determine if the target table exists */
%let DBXRC = %sysfunc(exist(testlib."ADVERSE_SORTED"n, DATA));

%if &DBXRC>0 %then
%do; /* if table exists*/

```

```

/*-----
* Name: Truncate
* Description: Truncate a table
* Generated: Tue Jun 29 13:29:09 EDT 2008
*-----*/
%put NOTE: Truncating table ...;

/* get the constraints from the table */
proc contents data = testlib."ADVERSE_SORTED"n
out2 = work.etls_constraints
noprint;
run;

/* get the number of constraints (number of rows) */
%let etl_numRows = 0;
%let etl_dsid=%sysfunc(open(work.etls_constraints));
%if (&etl_dsid gt 0) %then
%do;
%let etl_numRows = %sysfunc(attrn(&etl_dsid, NOBS));
%let etl_dsid = %sysfunc(close(&etl_dsid));
%end;

%let etl_primaryKey = NO;

%if (&etl_numRows gt 0) %then
%do; /* table has constraints */

/* determine if another table has a foreign key that points to this table */
data work.etls_constraints;
set work.etls_constraints;
type = upcase(type);
if (type eq "REFERENTIAL") then
do;
call symput("etl_primaryKey", "YES");
stop;
end;

/* delete any indexes that are created by another constraint */
if (type eq "INDEX" and ICOwn eq "YES") then
delete;
run;

%end; /* table has constraints */

%if (&etl_primaryKey eq YES) %then
%do; /* table has primary key and referential constraints */

data _null_;
put "WARNING: Because the target table has referential integrity "
constraint(s), an attempt will be made to truncate the table using "
the 'delete&039: statement in sql. This procedure may fail if the "
constraints are violated. Note that if the procedure is successful,
the rows will only be logically deleted, not physically deleted.";
run;

```

```

/* logically delete all the records from the table */
proc sql;
delete * from testlib."ADVERSE_SORTED"n;
quit;

%RCSET(&sqlrc);

%end; /* table has primary key and referential constraints */

%else
%do; /* table does not have a primary key and referential constraints */

%if (&etl_numRows gt 0) %then
%do; /* table has constraints */

/* delete the constraints from the table */
proc datasets lib=testlib nolist;
modify "ADVERSE_SORTED"n;
ic delete _all_;
quit;

%end; /* table has constraints */

/* physically delete all the records from the table */
data testlib."ADVERSE_SORTED"n;
set testlib."ADVERSE_SORTED"n;
stop;
run;

%RCSET(&syserr);

%if (&etl_numRows gt 0) %then
%do; /* table has constraints */

/* recreate the constraints on the table */
data _null_;

set work.etls_constraints end=eof;

if _n_ eq 1 then
do;
call execute("proc datasets lib=testlib nolist;");
call execute(& modify "ADVERSE_SORTED"n;');
end;

call execute(" " || recreate);

if eof then
call execute("quit;");

run;

%RCSET(&syserr);

%end; /* table has constraints */

```



```
%end; /* table does not have a primary key and referential constraints */

%put NOTE: Deleting work.etls_constraints...;
proc datasets lib=work nolist nowarn memtype=(data view);
delete etls_constraints;
quit;

%end; /* if table exists*/

/*-----
* Name: Create Table
* Description: Create a new table
* Generated: Tue Jun 29 13:29:09 EDT 2008
*-----*/
%if &DBXRC=0 %then
%do; /* if table does not exist*/

%put NOTE: Creating table ...;

data testlib."ADVERSE_SORTED"n
(label="ADVERSE2");
attrib "aedeod"n length=$21 format=$F21. informat=$F21.
label="AE Decode from Dictionary";
attrib "subjid"n length=8 format=BEST12. informat=F12.
label="Subject ID";
attrib "studyid"n length=$8 format=$F8. informat=$F8.
label="Study ID";
attrib "trtgrp"n length=$8 format=$F8. informat=$F8.
label="Treatment Group";
attrib "bodsys"n length=$20
label="Body System";
attrib "aesev"n length=$10
label="Severity";
attrib "aeout"n length=$15< br> label="Outcome";
stop;
run;

%RCSET(&syserr);

%end; /* if table does not exist*/

%let sqlrc = 0;
/*-----
* Name: Append
* Description: Append new data
* Generated: Tue Jun 29 13:29:09 EDT 2008
*-----*/
%put NOTE: Appending data ...;

proc append base=testlib."ADVERSE_SORTED"n
data=&DBXLAST (&SYSOPT) force;
run;
%RCSET(&syserr);

%mend dbwaload;
```

```
/*-----  
* Name: DBWALOAD  
* Description: Execute load data macro  
* Generated: Tue Jun 29 13:29:09 EDT 2008  
*-----*/  
%dbwaload;
```

### **Macro Variables for Status Handling in User-Written Code**

You can add the RCSET macro and the TRANS\_RC and JOB\_RC variables to user-written code, such as the code for the User Written Code transformations and generated transformations. Use the preceding example as a model for your code.

## Chapter 9

# Deploying Jobs

---

<b>About Deploying Jobs</b> .....	<b>184</b>
<b>About Deploying Jobs for Scheduling</b> .....	<b>185</b>
<b>Prerequisites for Deploying a Job for Scheduling</b> .....	<b>185</b>
<b>Deploying Jobs for Scheduling</b> .....	<b>185</b>
Problem .....	185
Solution .....	185
Tasks .....	185
<b>Redeploying Jobs for Scheduling</b> .....	<b>187</b>
Problem .....	187
Solution .....	187
Tasks .....	187
<b>Using Scheduling to Handle Complex Process Flows</b> .....	<b>187</b>
Problem .....	187
Solution .....	188
Tasks .....	188
<b>Using Deploy for Scheduling to Execute Jobs on a Remote Host</b> .....	<b>188</b>
Problem .....	188
Solution .....	188
Tasks .....	189
<b>About Deploying Jobs as Stored Processes</b> .....	<b>189</b>
<b>Prerequisites for Deploying a Job as a Stored Process</b> .....	<b>190</b>
For Administrators .....	190
For Users .....	190
<b>Deploying Jobs as Stored Processes</b> .....	<b>190</b>
Problem .....	190
Solution .....	190
Tasks .....	190
<b>Redeploying Jobs to Stored Processes</b> .....	<b>192</b>
Problem .....	192
Solution .....	192
Tasks .....	193
<b>Viewing or Updating Stored Process Metadata</b> .....	<b>193</b>
Problem .....	193
Solution .....	193
Tasks .....	194

<b>About Deploying Jobs as Web Services</b> .....	<b>194</b>
<b>Prerequisites for Web Service Jobs</b> .....	<b>195</b>
For Administrators .....	195
For Users .....	195
<b>Requirements for Web Service Jobs</b> .....	<b>195</b>
<b>Creating a Web Service Job</b> .....	<b>196</b>
Problem .....	196
Solution .....	196
Tasks .....	197
<b>Deploying a Web Service Job as a Stored Process</b> .....	<b>200</b>
Problem .....	200
Solution .....	200
Tasks .....	201
<b>Deploying a Stored Process as a Web Service</b> .....	<b>203</b>
Problem .....	203
Solution .....	203
Tasks .....	203

---

## 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 Deploying Jobs for Scheduling” on page 185](#).
- Deploy a job as a SAS stored process; see [“About Deploying Jobs as Stored Processes” on page 189](#).
- Deploy a job as a SAS stored process that can be accessed by a Web service client; see [“About Deploying Jobs as Web Services” on page 194](#).

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 187](#). Alternatively, you can drop jobs into other jobs, and build up complexity that way as well. For example, you could build an outer job that contains inner jobs. You might find that these nested jobs provide a more direct and efficient solution to the problem of creating and scheduling complex process flows. This approach does not require separate deployment steps. For more information, see [“Creating a Job That Contains Jobs” on page 125](#).
- Execute a job on a remote host; see [“Using Deploy for Scheduling to Execute Jobs on a Remote Host” on page 188](#). Alternatively, you can save the SAS code generated by the job to a file, and then manually move that file to the remote host.

*Note:* Under change management, only administrators can deploy jobs.

---

## About Deploying Jobs for Scheduling

You can select a job in the Inventory tree or the Folders tree and deploy it for scheduling. Code is generated for the job, and the code is saved to a file in a source repository. Metadata about the deployed job is saved to the current metadata server. The user or administrator responsible for scheduling jobs can use the appropriate software to schedule the job for execution.

Here are some of the main tasks that are associated with deploying a job for scheduling:

- [“Deploying Jobs for Scheduling” on page 185](#)
- [“Redeploying Jobs for Scheduling” on page 187](#)
- [“Using Scheduling to Handle Complex Process Flows” on page 187](#)

See also [“Prerequisites for Deploying a Job for Scheduling” on page 185](#).

---

## Prerequisites for Deploying a Job for Scheduling

Administrators must install and configure a SAS Workspace Server for deploying jobs for scheduling. For more information, see *Scheduling in SAS*. The administrator then tells SAS Data Integration Studio users which server and deployment directory to select when deploying jobs for scheduling.

---

## 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:

- Use SAS Data Integration Studio to deploy the job for scheduling. See [“Deploy a Job for Scheduling” on page 185](#).
- Use other software to schedule the job for execution. For more information, see *Scheduling in SAS*.

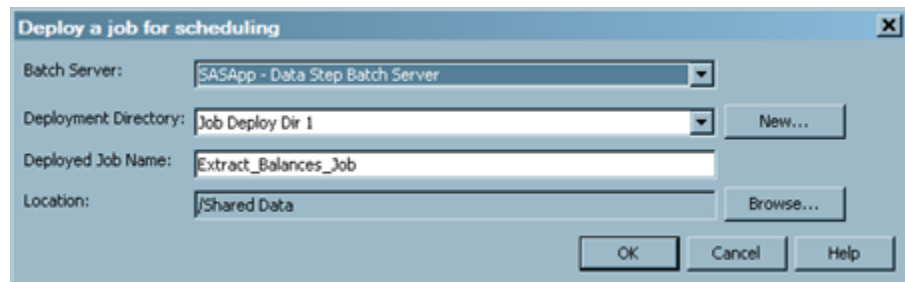
### Tasks

#### ***Deploy a Job for Scheduling***

Perform the following steps to deploy a job for scheduling:

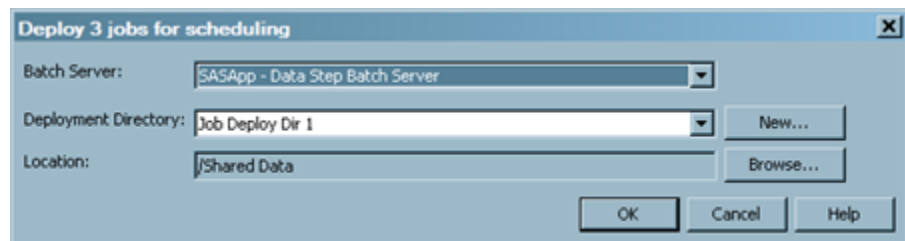
1. Right-click the job that you want to deploy. Then, select **Scheduling** ⇒ **Deploy** in the pop-up menu to access the Deploy for a job for scheduling window. The following display shows the window if you select only one job for deployment.

**Display 9.1** Deploy for a Job for Scheduling Window for a Single Job



By default, the deployed job file (in this case, Extract Balances Job.sas) is named after the selected job. The following display shows the Deploy for a job for scheduling window used to deploy multiple jobs for scheduling.

**Display 9.2** Deploy for Scheduling Window for Multiple Jobs



2. When you deploy more than one job, a separate SAS file is created for each job that you select. Each deployed job file is named after the corresponding job.
3. In the **Batch Server** field, accept the default server or select the server that is used to store the SAS file 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.
4. Check the **Deployment Directory** field to ensure that the deployed job is stored in the appropriate directory. If the wrong directory is displayed, select another director from the drop-down list, or click **New** to create a new directory if you have permission to create directories on the server.
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 **Deployed Job Name** field of the Deploy for a job for scheduling window. The name must be unique in the context of the directory specified in the **Deployment Directory** field.
6. 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 **Deployment Directory** field. Metadata about the deployed jobs is saved to the current SAS Metadata Server. A status window is displayed and indicates whether the deployment was successful. In the Inventory tree, metadata for the deployed job is added to the **Deployed job** folder. This job is now available for scheduling.

---

## Redeploying Jobs for Scheduling

### Problem

After a job has been deployed for scheduling, either the job or the computing environment changes. For example, additional transformations might be added to the process flow for the job, or the job might be exported to another environment where the servers and libraries are different.

### 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 187](#).
- Use other software to schedule the job for execution. For more information, see *Scheduling in SAS*.

### Tasks

#### **Redeploy a Job for Scheduling**

Perform the following steps to redeploy a job for scheduling:

1. Select **Tools** ⇒ **Redeploy for Scheduling** in the menu bar. Any jobs that have been deployed are found.
2. Click **Yes** to continue the redeployment process. The Redeployed scheduled jobs window is displayed. Verify that the appropriate options have been set, and click OK to redeploy the jobs. 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 environment. The regenerated jobs are now available for scheduling.

---

## 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 Folders tree. Perform the steps in [“Schedule Complex Process Flows” on page 188](#) to deploy and schedule the jobs in the proper sequence.

As an alternative to the approach described here, you can drop jobs into other jobs and build up complexity that way. For example, you can build an outer job that contains inner jobs. You might find that these nested jobs provide a more direct and efficient solution to the problem of creating and scheduling complex process flows. This approach does not require separate deployment steps. For more information, see [“Creating a Job That Contains Jobs” on page 125](#).

**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 Folders tree, and give the jobs a prefix that displays in the appropriate execution order.
3. Deploy the jobs for scheduling.
4. The user responsible for scheduling can use the appropriate software to schedule the jobs to be executed in the proper sequence.

---

## Using Deploy for Scheduling to Execute Jobs 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**

In order for this solution to work, a SAS Workspace Server and a SAS DATA Step Batch Server must have been configured on the remote host. For information about this configuration, administrators should see the "Multi-Tier Environments" section in the SAS Data Integration Studio chapter of the *SAS Intelligence Platform: Desktop Application Administration Guide*. Note especially the "Processing Jobs Remotely" topic.

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 SASApp, but you want a job to execute on another SAS Application Server that is called SASApp2. Select SASApp2 in the Deploy for Scheduling window, so that the code that is generated for the job is local to SASApp2.

*Note:* Instead of using this deployment mechanism, you can also save the SAS code generated by the job to a file. Then, you can move that file to the remote host.

## Tasks

### **Deploy One or More Jobs for Execution on a Remote Host**

Perform the following steps to deploy jobs for execution on a remote host:

1. In a tree view, right-click the job or jobs that you want to deploy. Then, select **Scheduling** ⇒ **Deploy** in the pop-up menu to access the Deploy for a job for scheduling window.
2. In the **Batch Server** field, select the SAS Application Server that contains the servers on the remote host.
3. In the **Deployment Directory** field, select a predefined directory where the generated code for the selected job is stored. If the wrong directory is displayed, click **New** and specify the correct directory in the New directory window.

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 **Deployed Job 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 **Deployment Directory** field. Metadata about the deployed jobs is saved to the current metadata server. In the Inventory tree, metadata for the deployed job is added to the **Deployed job** folder. The deployed job can either be scheduled or executed by any convenient means.

---

## About Deploying Jobs as Stored Processes

You can select a job in the Inventory tree or the Folders tree and deploy it as a SAS stored process. Code is generated for the stored process and the code is saved to a file in a source repository. Metadata about the stored process is saved to the current metadata server. The stored process 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.

Here are some of the main tasks that are associated with deploying a job as a stored process:

- “Deploying Jobs as Stored Processes” on page 190

- “Redeploying Jobs to Stored Processes” on page 192
- “Viewing or Updating Stored Process Metadata” on page 193

See also “Prerequisites for Deploying a Job as a Stored Process” on page 190.

---

## Prerequisites for Deploying a Job as a Stored Process

### ***For Administrators***

The New Stored Process wizard requires a connection to a server that can execute SAS stored processes. Administrators install and configure the appropriate servers, and then tell SAS Data Integration Studio users which server and source repository to select when deploying jobs as 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 on a SAS Metadata Server, see *SAS Intelligence Platform: Application Server Administration Guide*.

### ***For Users***

To use the stored process feature efficiently, you should be familiar with stored process parameters, input streams, and result types. For a detailed discussion of stored processes, see *SAS Stored Processes: Developer's Guide*.

---

## Deploying Jobs as Stored Processes

### ***Problem***

You want to make a job available to any application that can execute a SAS stored process.

### ***Solution***

Deploy 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 Folders tree on the SAS Data Integration Studio desktop, right-click the job for which you want 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 9.3** General Tab

**New Stored Process**

**Stored Process Wizard : General Tab**

Specify the name, description and keywords for the New Stored Process

Name:

Description:

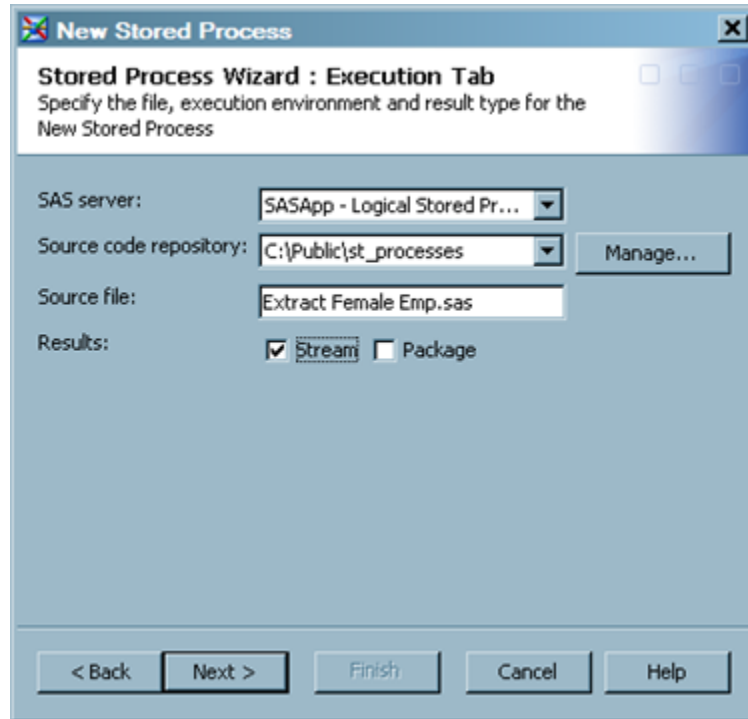
Keywords:

Responsibilities:

Name	Role

< Back   Next >   Finish   Cancel   Help

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**. Click **Next** to access the **Execution** tab of the wizard.
3. 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 9.4** Execution Tab

Click **Next** to access the Parameters Tab screen, where you can specify any parameters that you need for the stored process.

4. Click **Next** to access the Data screen, where you can specify any data sources and targets that are used by the stored process.
5. Click **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 metadata server. A metadata object for the stored process is added to the **Stored Process** folder in the Inventory tree.

After the job has been deployed, it can be executed with any application that can execute a SAS stored process.

---

## Redeploying Jobs to Stored Processes

### Problem

After a job has been deployed as a stored process, either the job or the computing environment changes. For example, additional transformations might be added to the process flow for the job, or the job might be exported to another environment where the servers and libraries are different.

### 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 193](#).

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 193](#).

## 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 to access Redeploy Jobs to Stored Processes window.
4. Click **Yes**.

### **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** to access the Redeploy Jobs to Stored Processes window.
2. Click **Yes**.

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 SAS Metadata Server.

---

## 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.

## 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 as Web Services

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.

After a SAS Data Integration Studio job has been deployed as a stored process, you can select the stored process in the Inventory tree or the Folders tree and deploy it as a Web service. Code is generated for the Web service and the code is saved to a file in a source repository. Metadata about the Web service is saved to the current metadata server. The Web service can be executed as required by a Web service client.

To deploy a job as a Web service, perform the following tasks:

- Create the job. See [“Creating a Web Service Job” on page 196](#).
- Deploy the job as a stored process. See [“Deploying Jobs as Stored Processes” on page 190](#).
- Deploy the stored process for execution by a Web service client. See [“Deploying a Stored Process as a Web Service” on page 203](#).

After the job has been deployed, the user responsible for executing the deployed job can use the appropriate Web service client to access and execute the job. Before deploying a job as a Web service, you might want to review the general prerequisites that are described in [“Prerequisites for Web Service Jobs” on page 195](#) and the specific requirements that are described in [“Requirements for Web Service Jobs” on page 195](#).

---

## Prerequisites for Web Service Jobs

### For Administrators

To deploy a job as a Web service, users must first deploy the job as a stored process. Accordingly, the prerequisites that are described in [“Prerequisites for Deploying a Job as a Stored Process” on page 190](#) must be met.

The Deploy as a Web Service wizard requires a URL to a Web Service Maker. This URL is available when administrators have installed one of the following:

- SAS BI Web Services for .NET, which is part of SAS Integration Technologies
- SAS Web Infrastructure Platform (WIP) and its associated components, which is included in the BI Server and EBI Server software

### For Users

To use the Web service feature efficiently, you should be familiar with stored processes, XML tables, SAS XML libraries, Web services, and Web service clients. For more information about SAS XML libraries, see the *SAS XML LIBNAME Engine: User's Guide*.

---

## Requirements for Web Service Jobs

A Web service job is a SAS Data Integration Studio job that is designed to be executed by a Web service client. The process flow for a Web service job has these requirements:

- The job can receive zero or more inputs from the Web service client that executes the job.
- The job can send zero or one output to the client that executes the job.
- Input to the job from the client, and output from the job to the client, must be in XML table format.
- The XML tables that specify client input or output in the job must be members of a SAS XML library. For details about SAS XML libraries, see the *SAS XML LIBNAME Engine: User's Guide*.
- 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 XML table for each client input or output in the job must be configured as a Web stream.

The following display illustrates a typical process flow for a Web service job.

**Display 9.5** Sample Process Flow for a Web Service Job

In the sample flow, INTABLE is a metadata object for an input table in XML format. Convert Temp GT is a generated transformation with custom SAS code that processes the input. OUTTABLE is a metadata object for an output table in XML format. The small blue circle that overlays the table icons indicates that the input table and output table are configured as Web streams.

The preceding Web service job is deployed as a stored process. Then the stored process is deployed as a Web service. Users with Web client software access the Web service job, and they are prompted to supply input. The job processes the input and displays the result to the Web client.

---

## Creating a Web Service Job

### Problem

You want to create a job that can be executed by a Web service client. The job must be accessed across platforms, and the amount of data to be input and output is not large.

### Solution

Create a Web service job, deploy it as a stored process, and then deploy the stored process as a Web service.

Your first task is to create a Web service job. The job must meet the requirements that are described in [“Requirements for Web Service Jobs” on page 195](#). One way to meet these requirements is to create a job with a process flow similar to the flow in the following display.

**Display 9.6** Sample Process Flow for a Web Service Job

In the sample flow, INTABLE is a metadata object for an input table in XML format. Convert Temp GT is a generated transformation with custom SAS code that processes the input and produces a result. OUTTABLE is a metadata object for an output table in XML format. The small blue circle that overlays the table icons indicates that the input table and output table are configured as Web streams. Users with Web client software access the Web service job, and they are prompted to supply input. The job processes the input and displays the result to the Web client.

To create a Web service job, perform the following tasks:

- [“Create the XML Inputs and Outputs for the Job ” on page 197](#)
- [“Create XML Libraries for the Inputs and Outputs” on page 197](#)
- [“Register the XML Inputs and Outputs” on page 198](#)



- “Create a Generated Transformation That Produces the Desired Output” on page 198
- “Create the Job” on page 200

It is assumed that the general prerequisites have been met, as described in “Prerequisites for Web Service Jobs” on page 195.

## Tasks

### Create the XML Inputs and Outputs for the Job

Perform the following steps to create the input and output tables for a Web service job. If you include test values in these tables, you might find it easier to test your job before it is deployed.

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 the sample job that is shown in [Sample Process Flow for a Web Service Job on page 196](#), the physical name of the input table is InTemp.xml. The XML code for this table is as follows:

```
<TABLE>
  <INTABLE>
    <temperature> 40 </temperature>
    <Unit> C </Unit>
  </INTABLE>
</TABLE>
```

2. Use an XML editor to create an XML table for the output to the Web service client. Save that table to a file. For the sample job, the physical name of the output table is OutTemp.xml. The XML code for this table is as follows:

```
<TABLE>
  <OUTTABLE>
    <CalculatedTemperature> Temperature of 40 degrees Centigrade = 104 degrees F
  </OUTTABLE>
</TABLE>
```

### Create XML Libraries for the Inputs and Outputs

You must create a separate XML library for each input from the Web service client and each output from the job. SAS XML libraries differ from most SAS libraries in that the library metadata points to an XML file, not to a directory that contains XML files. The structure of your XML tables might require you to specify certain options in the library. For details about SAS XML libraries, see the *SAS XML LIBNAME Engine: User's Guide*.

Perform the following steps to create the libraries for the input and output tables in a Web service job:

1. On a file system that is accessible to the Web service client, create directories for the input and output tables. For the sample job, the physical path of the input directory is c:\public\input. The physical path of the output directory is c:\public\output.
2. Copy the input and output files that you created to the directories that you created. For the sample job, the physical path of the input file is c:\public\input\InTemp.xml. The physical path of the output file is c:\public\output\OutTemp.xml.
3. In SAS Data Integration Studio, to register a library for an input table in XML format, right-click a destination folder in the Folders tree. Then select **New** ⇒ **Library** from the pop-up menu.

4. In the New Library wizard, select **SAS XML Library** and click **Next**.
5. Use the pages of the wizard to specify values that are appropriate for the library for the input table. For the sample job, you can enter the following values:

**Name:** *Intemp*

**Selected Server:** *SASApp*

**Libref:** *intemp*

**Engine:** *XML*

**XML File:** *c:\public\input\InTemp.xml*

**XML Type:** *Generic*

**Library Access:** *Blank*

6. Repeat steps 1 through 5 for the output library. Use the pages of the wizard to specify values that are appropriate for that library. For the sample job, you can enter the following values:

**Name:** *Outtemp*

**Selected Server:** *SASApp*

**Libref:** *outtemp*

**Engine:** *XML*

**XML File:** *c:\public\output\OutTemp.xml*

**XML Type:** *Generic*

**Library Access:** *Blank*

### **Register the XML Inputs and Outputs**

Perform the following steps to register the input and output tables for a Web service job:

1. Right-click the input library and click **Register Tables** in the pop-up menu.
2. Register the input table. For the sample job, the input table is InTemp.xml. For more information, see [“Register a Table with the Register Tables Wizard” on page 65](#).
3. Right-click the output library and click **Register Tables** in the pop-up menu.
4. Register the output table. For the sample job, the output table is OutTemp.xml.

### **Create a Generated Transformation That Produces the Desired Output**

You can use the Transformation Generator wizard to create a custom transformation that reads input in the form of an XML table, process the input, and then write output in the form of an XML table. For an introduction to the Transformation Generator wizard, see [“Creating and Using a Generated Transformation” on page 219](#).

In the sample job, we need a custom transformation that reads values for temperature and scale, in the format specified by InTemp.xml. The transformation converts the temperature in one scale to the equivalent temperature in the other scale, and then writes the result in the format specified by OutTemp.xml.

Perform the following steps or similar steps to create a custom transformation for a job that can be deployed as a Web service:

1. Right-click the destination folder in the Folders tree where the new transformation should be stored. Then select **New** ⇒ **Transformation**. The first page of the Transformation Generator wizard displays.
2. Enter a name for the transformation. In the sample job, the transformation is named **Convert Temp GT**.
3. Review other values on this page and make changes as desired, and then click **Next**. The SAS Code page displays.
4. Add SAS code that reads input in the form of an XML table, process the input, and then write output in the form of an XML table. In the sample job, the following SAS code is added to this page.

```
data &_OUTPUT;
set &_INPUT;
keep CalculatedTemperature;
length NewTemperature 8.;
if (Unit="F") then
do;
    NewTemperature=(5/9)*(Temperature-32);
    Unit="C";
    CalculatedTemperature = "Temperature of " || compress(temperature) ||
        " degrees Farenheit = " || compress(NewTemperature) ||
        " degrees Centigrade" ;
end;
else if (Unit="C") then
do;
    NewTemperature=(9/5)*(Temperature)+32;
    Unit="F";
    CalculatedTemperature = "Temperature of " || compress(temperature) ||
        " degrees Centigrade = " || compress(NewTemperature) ||
        " degrees Farenheit" ;
end;
else
do;
    CalculatedTemperature="Temperature of " || compress(temperature) ||
        " with unit of " || compress(unit) || " cannot be converted ";
    Unit="";
end;
run;
```

5. When you are satisfied with the code, click **Next**. The Options page displays. Specify options as desired. The sample job does not require any options. When ready, click **Next**. The Transform properties page displays.
6. Specify transformation properties as desired. For the sample job, the following properties are specified:

**Transform supports inputs** (selected)

**Maximum number of inputs** (1)

**Transform supports outputs** (selected)

**Maximum number of outputs** (1)

**Automatically generate delete code for outputs** (deselected)

*Note:* Be sure to deselect the **Automatically generate delete code for outputs** property. It is not appropriate for Web service jobs.

- Click **Finish** to save the transformation. In the Folders tree, the custom transformation appears in the folder that you right-clicked in step 1. In the Transformations tree, the custom transformation appears in the **Ungrouped** folder or another category that you specified in step 3.

### Create the Job

Perform the following steps to create the process flow for a job that can be deployed as a Web service:

- Right-click the destination folder in the Folders tree where the new job should be stored. Then select **New** ⇒ **Job**. The New Jobs wizard displays.
- Enter a name for the job. The sample job is named **Convert Temp Job**. Click **OK**. An empty job opens in the Job Editor.
- Drag your custom transformation from a tree view into the job.
- Drag an XML input table from a tree view into the job. Connect the input to the custom transformation. Repeat for as many inputs as you have.
- Right-click the temporary output table for the transformation and select **Replace**. Select the XML output table.

*Note:* At this point, you should have a complete process flow. The process flow for the sample job looks similar to the process flow shown in the [Sample Process Flow for a Web Service Job display on page 196](#).

- 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 note the status messages. You can right-click the output table and select **Open** to verify that the values in the client output table are correct. If not, troubleshoot and correct 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.

- 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. The Web stream icon, a small blue circle, should overlay the table icons for all tables in the job.
- Save and close the job.

---

## Deploying a Web Service Job as a Stored Process

### Problem

You want to deploy a Web service job as a stored process so that the stored process can be deployed as a Web service.

### Solution

Use the New Stored Process wizard to deploy a Web service job as a stored process.

## Tasks

### ***Deploy a Web Service Job as a Stored Process***

Perform the following steps to deploy a Web service job as a stored process:

1. Right-click the Web service job in a tree view, and select **Stored Process** ⇒ **New** from the pop-up menu. The New Stored Process wizard displays.
2. Accept the default name or specify another name that makes it easier to distinguish the job from the stored process that you are about to create. For the sample job, the name is **Convert Temp Stp**. Enter other values as desired and click **Next**. The Execution page displays.
3. Verify that the values in the following fields are appropriate. If not, select an appropriate value.

**SAS Server** specifies the name of the SAS server that runs the stored process that you are defining. For the sample job, this is SAS App – Logical Stored Process Server.

**Source code repository** specifies the path where the SAS server saves the source code for the stored process. For the sample job, this path is c:\public\st\_processes.

**Source file** specifies the name of the SAS file that contains the stored process that you are creating. For the sample job, this is Convert Temp Job.sas.

When ready, click **Next**. The Parameters page displays.

4. (Optional) Enter parameters if desired. The sample job does not require parameters. Click **Next** to go to the Data page.
5. The Data page shows information about the source and target in the job. Verify that the information on the Data page is appropriate for the stored process that you are creating. If not, use the **New** or **Edit** buttons to specify appropriate values for the source and target. For example, the following display shows the default information on the Data page for the sample job.

**Display 9.7** Data Page of the New Stored Process Wizard

**New Stored Process**

**Data**  
Specify data sources and targets used by the stored process.

Sources:

Label	Type	Fileref
INTABLE	XMLStream	intemp

Buttons: New..., Edit..., Delete, Move Up, Move Down

Targets:

Label	Type	Fileref
OUTTABLE	XMLStream	outtemp

Buttons: New..., Edit..., Delete, Move Up, Move Down

Navigation: < Back, Next >, Finish, Cancel, Help

To update the source information, select the appropriate row in the Source pane, and then click **Edit**. A Modify Data Source window displays. For the sample job, you can specify values such as the following:

**Type:** XML Stream

**Label:** Input Temperature and Unit

**Allow rewinding stream:** (selected)

**Fileref:** intemp

**Specify schema:** (selected)

**Schema URI:** file:///c:/public/InTable.xsd

**Reference namespace:** http://server1/test (as specified in the schema)

**Reference name:** TABLE

**Reference type:** Schema element

**WSDL generation options:** embedded

To update the target information, select the appropriate row in the Target pane, and then click **Edit**. A Modify Data Target window displays. For the sample job, you can specify values such as the following.

**Type:** XML Stream

**Label:** Output Temperature

**Fileref:** outtemp

- Review any changes. Click **Finish** when ready. A stored process is generated for the job. A metadata object for the stored process is added to the **Stored Process** folder in the Inventory tree.

You might want to use an appropriate application to run the stored process to ensure that it works.

---

## Deploying a Stored Process as a Web Service

### **Problem**

You want to deploy a stored process as a Web service, so that it can be executed by a Web service client.

### **Solution**

Use the Deploy As Web Service wizard to deploy a stored process as a Web service. Typically, the stored process is created from a Web service job, as described in [“Deploying a Web Service Job as a Stored Process” on page 200](#).

### **Tasks**

#### ***Deploy a Stored Process as a Web Service***

Perform the following steps to deploy a stored process as a Web service:

1. Right-click the stored process in a tree view and select **Web Service** ⇒ **New** from the pop-up menu. The Deploy As Web Service wizard displays.
2. Select a URL for the Web Service Maker. If you do not see a URL, contact your administrator.
3. Specify a name for the Web service. Slashes, backslashes, spaces, and control characters cannot be used in this field.
4. Typically the **Use my current credentials to deploy** check box should be selected. When ready click **Next**. The Namespace and Keywords page displays.
5. If the defaults are acceptable, click **Next**. The Confirm Web Service Deployment page displays.
6. If the defaults are acceptable, click **Finish**. A Web service is generated. If the operation is successful, a dialog box is displayed. Click **OK** to close it. A metadata object is added to the **Web service (generated)** folder in the Inventory tree.

After the stored process has been deployed as a Web service, it can be executed with a Web service client.





## Chapter 10

# Working with Generated Code

---

<b>About Code Generated for Jobs</b> . . . . .	<b>205</b>
Overview . . . . .	205
LIBNAME Statements . . . . .	206
SYSLAST Macro Statements . . . . .	206
Remote Connection Statements . . . . .	207
Macro Variables . . . . .	207
User Credentials in Generated Code . . . . .	208
<b>Displaying the Code Generated for a Job</b> . . . . .	<b>209</b>
Problem . . . . .	209
Solution . . . . .	209
Tasks . . . . .	209
<b>Displaying the Code Generated for a Transformation</b> . . . . .	<b>209</b>
Problem . . . . .	209
Solution . . . . .	209
Tasks . . . . .	210
<b>Specifying Options for Jobs</b> . . . . .	<b>210</b>
Problem . . . . .	210
Solution . . . . .	210
Tasks . . . . .	210
<b>Specifying Options for a Transformation</b> . . . . .	<b>210</b>
Problem . . . . .	210
Solution . . . . .	211
Tasks . . . . .	211
<b>Modifying Configuration Files or SAS Start Commands for Application Servers</b> . . . . .	<b>211</b>

---

## 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 210](#) and [“Specifying Options for a Transformation” on page 210](#).

## 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 <engine> <"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 <library details>;
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 variable to hold the name of the transformation's output table. In general, 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.

## Remote Connection Statements

Most transformations within a job can specify their 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 is done implicitly for users if the machine is remote. Users can also use the Data Transfer transformation to explicitly handle moving data between machine environments when needed. The Data Transfer transformation provides more control over the transfer when needed, such as support for locale-specific settings.

## Macro Variables

When SAS Data Integration Studio generates the code for a job, the code includes the macro variables that are listed in the following table:

Macro Variable	Description
etls_jobName	Specifies the name as supplied on the job properties panel.
etls_userID	Specifies the user ID that is used to generate the code for the job.
_INPUT	Specifies the libref.tablename of the first input table.
_INPUT_count	Specifies the count of input tables.
_INPUT_connect	Specifies the connect statement for the table. This macro variable is used for explicit passthrough statements.
_INPUT_engine	Specifies the library engine. This macro variable can be used for explicit passthrough statement construction.
_INPUT_memtype	Specifies the member type of the table, either DATA or VIEW. Users can use this variable to write transformation code to enable creation of views on output tables or to know whether the input is a VIEW.

Macro Variable	Description
<code>_INPUT_options</code>	Specifies the table option string, such as COMPRESS=YES ENCRYPT=YES. This macro option is found on the table options dialog box from physical storage tab on the table's Properties window.
<code>_INPUT_alter</code>	Specifies an alter or password option text so the table can be deleted or altered. This macro variable is a subset of the <code>_options</code> string.
<code>_INPUT_path</code>	Specifies the location of table on metadata server.
<code>_INPUT_type</code>	Specifies a macro given by the prompting framework. This macro variable should always be 1 for usage with SAS Data Integration Studio.
<code>jobID</code>	Specifies the unique metadata ID code that is given to the job when the job is first created.
<code>JOB_RC</code>	Specifies a status handling macro variable that is set and reset (as the job runs) to be the maximum return code value ( <code>&amp;trans_rc</code> ) of the completed transformations.
<code>_OUTPUT_count</code>	Specifies the count of output tables.
<code>SYSLAST</code>	Specifies the name of the transformation's output table. In general, 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.
<code>trans_rc</code>	Specifies a status handling macro variable that is set based on the return code of individual steps within a transformation.

*Note:* Any variable that begins with `_INPUT` or `_OUTPUT` deals with the macros that are always generated with transformations that have inputs, outputs, or both. `_Input` and `_output` are present on the first table by default because SAS Data Integration Studio uses a legacy macro set. If identical `_INPUT` and `_INPUT1` variables are present, `_INPUT1` is the name that the user chose when setting up the INPUT macro variable or the default if a name wasn't specified for the macro.

Users can add references to any of these in user-written code. See [“About User-Written Code” on page 213](#). SAS Data Integration Studio uses these macro variables in header comments and in code that is associated with the status handling features of the Return Code Checker, SQL Join, and loader transformations.

### User Credentials in Generated Code

The code that is generated is based on the credentials and permission settings of the user who 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, then the deployed job fails to execute. The solution is to redeploy the job with the appropriate credentials.

---

## 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 Job Editor window and selecting the **Code** tab. You can also view the SAS Code in the properties window for an unopened job. Note that 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.

### **Tasks**

#### ***View Code Displayed in the Job Editor Window***

To view the code for a job that is currently displayed in the Job Editor window, click the **Code** tab. The generated code for the job is displayed on the **Code** tab.

#### ***View Code for a Job Not Displayed in the Job Editor Window***

Perform the following steps to view the code for a job that is not displayed in the Job Editor window:

1. Right-click the job. Then, click **Properties** in the pop-up menu to open the properties window for the job.
2. Click the **Code** tab to display the generated code for the job.

---

## Displaying the Code Generated for a Transformation

### **Problem**

You want to see the code that you generated for a transformation.

### **Solution**

You can review the code for a transformation on the **Code** tab in the properties window for the transformation.

## Tasks

Perform the following steps to see the generated code for a transformation:

1. Open the properties window for the transformation.
2. Click the **Code** tab. The code that is generated for the transformation is displayed. The value in the **Code generation mode** field defaults to **Automatic**, which displays both the generated code for the transformation and the wrapper code that places it into the job. If you want to see the generated code for the transformation without the wrapper code, click **View Step Code**.

---

## Specifying Options for Jobs

### Problem

You want to set code generation options for SAS Data Integration Studio jobs, such as enabling parallel processing and configuring grid processing.

### Solution

In most cases the appropriate code generation options are selected by default, but you can override the default options. Use the **Code Generation** tab in the Options window to set global options for all new jobs. Use the **Options** tab in the properties window for a job to set local code generation options for that job.

## Tasks

### Set Global Options for Jobs

Use the **Code Generation** tab in the Options window to set global options for all new jobs. To display the tab, select **Tools** ⇒ **Options** ⇒ **Code Generation** from the menu bar. Then, specify the desired options.

### Set Local Options for a Job

Use the **Options** tab in the properties window for a job to set local options for that job. Right-click a job and select **Properties** to display the properties window. Click the **Options** tab. Set the appropriate options. These local options override global options for the selected job, but they do not affect any other jobs.

---

## Specifying Options for a Transformation

### Problem

You want to set options for a SAS 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. These options are described in [“Optimizing Sort Performance” on page 279](#).

---

## Modifying Configuration Files or SAS Start Commands for Application Servers

There are several ways to customize the environment where the code generated by SAS 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. Perform the following steps to invoke the ETLs\_DEBUG option in the autoexec.sas:

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:**

It is strongly recommended that you do not turn off MPRINT in a development environment.



## Chapter 11

# Working with User-Written Code

---

<b>About User-Written Code</b> .....	<b>213</b>
<b>Adding User-Written Code to the Precode and Postcode Tab</b> .....	<b>214</b>
Problem .....	214
Solution .....	214
Tasks .....	214
<b>Adding a User Written Code Transformation to a Job</b> .....	<b>216</b>
Problem .....	216
Solution .....	216
Tasks .....	216
<b>Creating and Using a Generated Transformation</b> .....	<b>219</b>
Problem .....	219
Solution .....	219
Tasks .....	220
<b>Maintaining a Generated Transformation</b> .....	<b>226</b>
Problem .....	226
Solution .....	226
Tasks .....	226
<b>Editing the Generated Code for a Job or Transformation</b> .....	<b>228</b>
Problem .....	228
Solution .....	228
Tasks .....	228
<b>Replacing the Generated Code for a Job or Transformation</b> .....	<b>229</b>
Problem .....	229
Solution .....	229
Tasks .....	229

---

## About User-Written Code

By default, SAS Data Integration Studio uses the metadata for a job to generate code for the job. If the generated code does not do what you want, you can do the following:

- add user-written code that will be executed before or after a job or transformation
- add a User-Written Code transformation to a job
- use the Transformation Generator wizard to create a custom transformation and add it to a job

- edit the generated code for a job or transformation
- replace the generated code for a job or transformation

---

## Adding User-Written Code to the Precode and Postcode Tab

### Problem

You want to set a SAS option, assign a libref, or perform some other action immediately before or after a job or transformation is executed.

### Solution

You can add the user-written code on the **Precode and Postcode** tab in the properties window for a job or transformation. For example, you can add a libref to an existing job that enables you to use a table from an unregistered library, as in the following sample job.

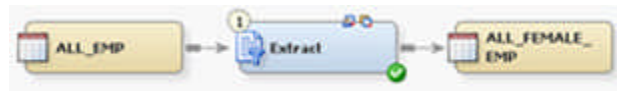
### Tasks

#### **Add the User-Written Code to the Precode or Postcode Field**

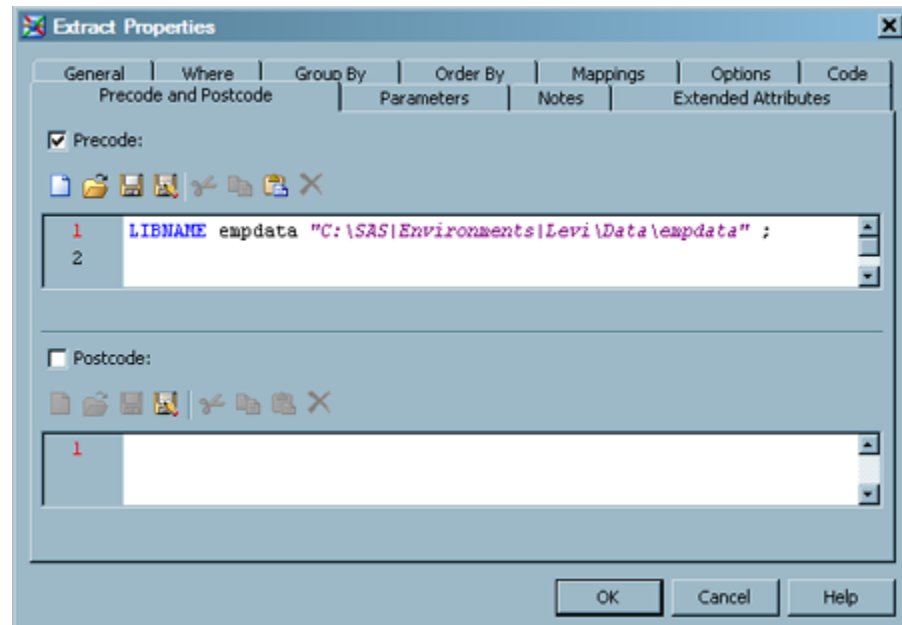
Perform the following steps to insert the user-written code:

1. Create a job, or open an existing job. The sample job, which is named is named Extract Job, is shown in the following display.

**Display 11.1** Sample Process Flow

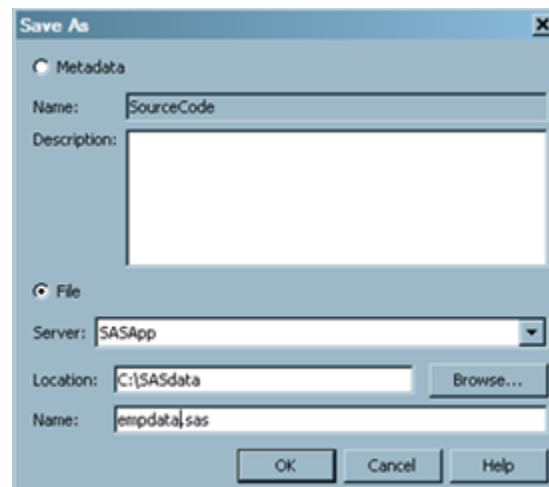


2. Open the **Precode and Postcode** tab in the properties window for the transformation or job that you need to change. In the sample job, the code is added to the job itself in order to provide access to the target table, ALL\_FEMALE\_EMP.
3. Select the appropriate **Precode** or **Postcode** check box. The check box that you select depends on whether the user-written code that you add runs before or after the source code for the job or transformation. The sample job requires precode.
4. Enter the user-written code in the field that is associated with the selected check box. The code shown in the following display is entered into the sample job.

**Display 11.2** Sample User-Written Precode**Save the User-Written Code to a File**

This is an optional task. Perform the following steps to save the user-written code to a file that you can reuse:

1. Click the **Save As** button to access the Save As window.
2. Select the **File** check box. Then, enter a server, name, and location for the file in the appropriate fields. The settings for the sample job are shown in the following display.

**Display 11.3** Sample Save As Window

*Note:* You can also select the **Metadata** check box and save the user-written code to the metadata server. In any case, the Save As window applies your changes to the current session. To make your changes persist after the current session, you must save the entire job. To save the entire job, select **File** ⇒ **Save** from the menu bar on the desktop.

3. Click **OK** to save the file and return to the properties window. Later, you can reuse the code in the file. Simply click the appropriate **Open** button on the **Precode and Postcode** tab.

4. Open the **Code** tab to verify that the user-written code is added to the job. The following display shows a portion of the **Code** tab for the sample job.

**Display 11.4** Sample Code Tab Content

```
/*---- Start of Pre-Process Code ----*/

LIBNAME expdata "C:\SAS\Environments\Lev2\Data\empdata";

/*---- End of Pre-Process Code ----*/
```

5. Click **OK** to save the changes to the job or transformation and close the properties window.

---

## Adding a User Written Code Transformation to a Job

### Problem

You want to add user-written code to a job. One method is to use the User Written Code transformation that is provided in Transformations tree. After you place this transformation in a job, you can add user-written code on the **Code** 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 that require only one input and output and no parameters. More complicated situations are handled more effectively with the Transformation Generator wizard.

### 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 need to map the columns from the transformation to the target table. Perform the following tasks:

- [“Create and Populate the Job” on page 216](#)
- [“Add User-Written Code to the User Written Code Transformation and Map Columns” on page 217](#)
- [“Run the Job” on page 218](#)
- [“View the Output” on page 218](#)

### Tasks

#### **Create and Populate the Job**

Perform the following tasks to create a job that uses the User Written Code transformation:

1. Create a new job and give it an appropriate name. The Job Editor window for the new job is displayed.
2. Drop the User Written Code transformation from the Data folder in the Transformations tree into the **Diagram** tab of the Job Editor window.
3. Connect the source table to the input port of the User Written Code transformation.

- Because you want a permanent target table to contain the output for the transformation, right-click the temporary work table that is attached to the transformation and click **Replace** in the pop-up menu. Then, use the Table Selector window to select the target table for the job. The target table must be registered in SAS Data Integration Studio. For more information about temporary work tables, see [“Working with Default Temporary Output Tables”](#) on page 126.

The flow for the sample job is shown in the following display.

**Display 11.5** Sample User Written Code Transformation in a Job



Note that the sample job includes a source table named EMP\_GENDER and a target table named CONVERTED\_EMP\_DATA.

### **Add User-Written Code to the User Written Code Transformation and Map Columns**

Perform the following steps to add user-written code to the User Written Code transformation in a job:

- 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 = "Male" then
    sex = "M";
  else if gender = "Female" then
    sex = "F";
  else
    sex="U";
run;
```

In this case, the code changes the gender identification in the Gender column from the words Male and Female to the initials M and F.

- Open the **Code** tab in the properties window for the User Written Code transformation on the **Diagram** tab of the Job Editor window. Code is generated for the transformation and displayed on the **Code** tab. The **Code generation mode** field defaults to **User written body**.
- Select the code generation mode. The **Code generation mode** field defaults to **User written body**. Note that any non-user-written portion of the code is dimmed when you select **User written body**. You cannot modify this part of the code.
- Place the cursor in an editable section of the **Code** tab.
- Enter the SAS code.
- Click **Save** or **Save As** on the toolbar for the tab. The **Save** option enables you to save the code in the editor as a metadata object (instead of saving the code into a file). The **Save As** option opens the Save File window, where you can either save a name and

description for the metadata object (code in the editor) or save the contents of the editor as a file.

*Note:* The **Save** and **Save As** options apply your changes to the current session. To make your changes persist after the current session, you must save the entire job.

To save the entire job, select **File** ⇒ **Save** from the menu bar on the desktop.

7. Click **OK** to save the changes and close the properties window.
8. Make sure that the User Written Code transformation is selected on the **Diagram** tab of the Job Editor window. Then, click the **Mappings** tab in the Details section.
9. Create column mappings between the source table and the target table.

*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. This mapping is also needed for impact analysis.

At this point, you have updated the User Written Code transformation so that it can retrieve the appropriate code when the job is executed.

### **Run the Job**

Perform the following steps to submit and run the job:

1. 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.
2. 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 11.6** Output from the Sample Job

#	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
17	Carol	F	14	62.8	102.5
18	Barbara	F	13	65.3	98
19	Alice	F	13	56.5	84

Note that the Gender column in the source table has been mapped to the Sex column in the target. The words Male and Female in the Sex column have been replaced with M and F.

## Creating and Using a Generated Transformation

### Problem

You need a custom transformation that enables you to process multiple outputs or inputs, macro variables, and parameters.

### Solution

Use the Transformation Generator wizard to create a custom transformation. The wizard guides you through the steps of creating the transformation and registering it on the metadata server. The new transformation displays in the Transformations tree, where it is available for use in any job.

Perform the following tasks:

- [“Create a Generated Transformation” on page 220](#)
- [“Use a Generated Transformation in a Job” on page 223](#)

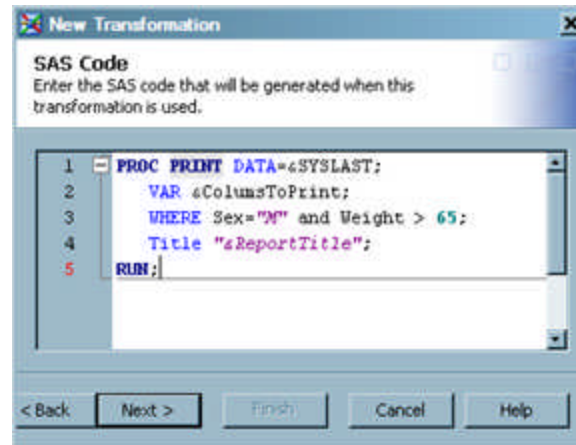
## Tasks

### Create a Generated Transformation

Perform the following steps to create a generated transformation:

1. Right-click the destination folder for the generated transformation.  
Then, select **New** ⇒ **Transformation** to access the Transformation Generator page in the New Transformation wizard.
2. Enter an appropriate name for the transformation. Then, verify that the destination folder for the transformation is populated in the **Location** field. You can also enter a description and select a category for the transformation. Click **Next** to access the SAS Code page.
3. Enter the SAS code generated by the transformation. 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 11.7** Sample Transformation Code Page



A number of macro variables appear in this sample code. One of these macro variables, `&SYSLAST`, is normally available and refers to the last data set created. The transformation also includes other macro variables, such as `&ColumnsToPrint` and `&ReportTitle`. The type of each such variable is defined in the Options screen of the wizard. You can supply values for these user-defined variables when the transformation is included in a job. Click **Next** to access the Options page.

4. Click **New Prompt** to access the New Prompt window. Define an option that corresponds to the first macro variable that is listed on the SAS code screen. The following display shows the **General** tab in the New Prompt window for the first macro variable in the sample transformation.



**Display 11.8** General Prompt Tab for the Columns to Print Option

The 'Edit Prompt' dialog box has two tabs: 'General' and 'Prompt Type and Values'. The 'General' tab is active. It contains the following fields:

- Name:** ColumnsToPrint
- Displayed text:** Columns to print
- Description:** Columns printed in report
- Parent group:** General (selected from a dropdown menu)
- Options:**
  - ☐ Hide from user
  - ☐ Requires a non-blank value
  - ☐ Read-only values

Buttons at the bottom: OK, Cancel, Help.

**Display 11.9** Prompt Type and Values Tab for the Columns to Print Option

The 'New Prompt' dialog box has two tabs: 'General' and 'Prompt Type and Values'. The 'Prompt Type and Values' tab is active. It contains the following settings:

- Prompt type:** Data source column (selected from a dropdown menu)
- Method for populating prompt:** User enters values (selected from a dropdown menu)
- Number of values:** Single value (selected from a dropdown menu)
- Columns to select from:**
  - ☒ Select from source
  - ☐ Select from target
- Data types:**
  - ☒ Character
  - ☒ Numeric
  - ☒ Date
  - ☒ Time
  - ☒ Timestamp
- ☐ Limit number of selectable columns
  - Minimum: 0
  - Maximum:
- ☐ Emit SQL syntax for columns

Buttons at the bottom: OK, Cancel, Help.

Each prompt window contains a **General** tab where you can enter general information about the option. Each prompt window also contains a **Prompt Type and Values** tab where you can select settings that are appropriate for each prompt type. For example, the second macro variable for the sample transformation, ReportType, requires an option that uses the text prompt type, as shown in the following display.

**Display 11.10** Sample Prompt Type and Value Tab for the ReportTitle Option

You need to define each of the macro variables that are 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 page for the sample transformation is depicted in the following display.

**Display 11.11** Completed Options Page

Displayed Text	Name	Type
General		Standard group
Columns to print	ColumnsToPrint	Data source column
Report title	ReportTitle	Text

When you have defined options for each of the macro variables, click **Next** to access the Transform properties page.

5. Use the Transform properties screen to specify the number of inputs and outputs for the generated transformation. The Transform properties page for the sample transformation is depicted in the following display.

**Display 11.12** Sample Transform Properties Page

These values determine how many inputs can be fed into the generated transformation. Note that 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 feature enables you to increase the minimum number of inputs without breaking existing jobs.

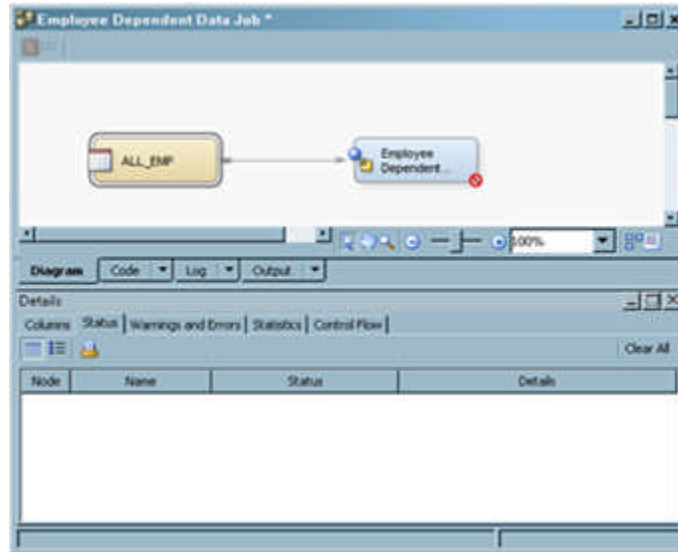
The increased maximum number of inputs is used to allow you to feed additional inputs into the transformation. (In the sample transformation, you can have up to six inputs because you set the maximum to six.) The same rules apply to outputs. The report that is generated by this transformation is sent to the **Output** tab of the Process Designer window. Therefore, you do not need to add an output to the transformation by using the controls in the **Outputs** group box.

6. Click **Next** to access the Finish page. Verify that the metadata is correct, and then click **Finish**. Your transformation is created and saved.
7. Verify that the generated transformation is available in the destination folder.

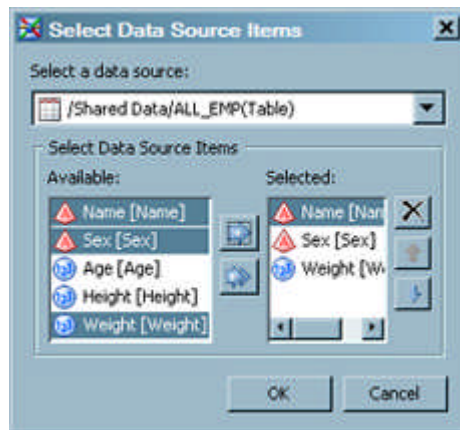
### **Use a Generated Transformation in a Job**

Perform the following steps to create and run a job that contains the generated transformation:

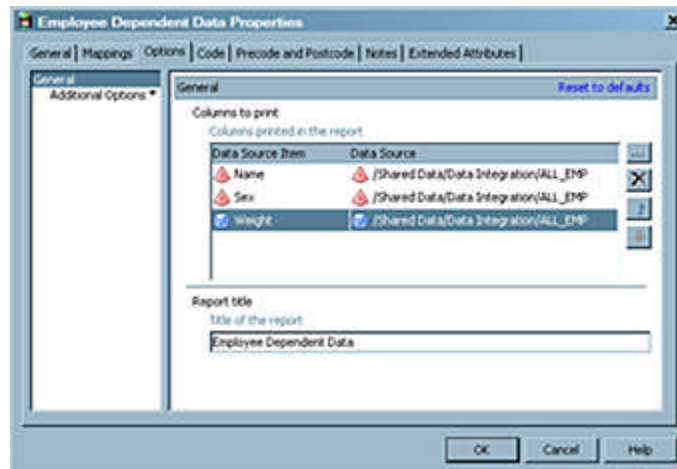
1. Create an empty job.
2. Drop the generated transformation into the Job Editor window for the empty job.
3. Drop the source table for the job into the Job Editor window.
4. If you enabled an output table, then drop the target table into the Job Editor window. You can also send the output to the **Output** tab of the Job Editor window. The appropriate option on the **General** tab of the Options window must be set so that the **Output** tab appears in the Job Editor window. The sample job shown in the following display uses the **Output** tab in this way.

**Display 11.13** Generated Transformation in a Sample Job

5. Drag the cursor from the output port of the transformation to the target table, if you have an output table. This action connects the transformation to the target.
6. Open the **Options** tab in the properties window for the generated transformation. Enter appropriate values for each of the options that are created for the transformation. Then, set the properties for the first option in the transformation. The following display shows the Select Data Source Items window, which is used to select the columns that are printed in the report.

**Display 11.14** Sample Select Data Source Items Window

The following display shows the completed **Options** tab.

**Display 11.15** Sample Completed Options Page

Note that the report title is already entered in the sample job. It was entered when the prompt was created.

Click **OK** to close the properties window and save the settings.

7. Run the job by right-clicking inside the Job Editor and selecting **Run** from the pop-up menu. SAS Data Integration Studio generates and runs the following code:

```
%let ColumnsToPrint = Name Sex Weight;
%let ColumnsToPrint_count = 3;
%let ColumnsToPrint0 = 3;
%let ColumnsToPrint1 = Name;
%let ColumnsToPrint2 = Sex;
%let ColumnsToPrint3 = Weight;
%let ReportTitle = %nrquote(Employee Dependent Data);
%let ColumnsToPrint_dsc = ;
%let GenerateIndexesOnTargets " " %nrquote(YES);
```

```
PROC PRINT DATA=&SYSLAST;
  VAR &ColumnsToPrint;
  WHERE Sex="M" and Weight > 65;
  Title "&ReportTitle;";
run;
```

8. After the code has executed, check the Job Editor window **Output** tab for the report that is shown in the following display.

**Display 11.16** 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	

---

## Maintaining a Generated Transformation

### Problem

You want to analyze the impact of a change to a generated transformations and perhaps update that transformation.

### Solution

Before you change a generated transformation, you should run impact analysis on that transformation to see all of the jobs that might 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 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 makes changes to a generated transformation that adversely affects many users.

Perform the following tasks:

- [“Identify a Generated Transformation” on page 226](#)
- [“Analyze the Impact of Generated Transformations” on page 226](#)
- [“Update Generated Transformations” on page 227](#)

### Tasks

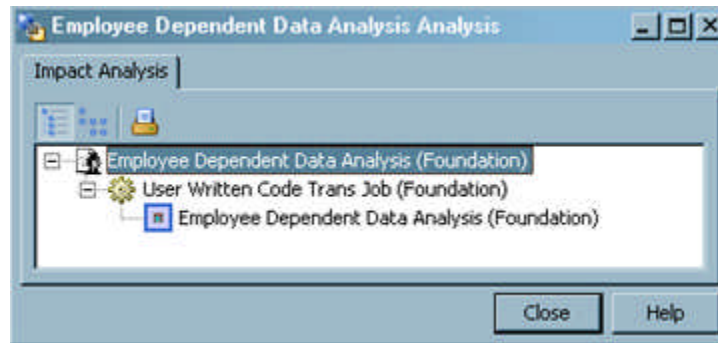
#### **Identify a Generated Transformation**

All transformations in the Transformation tree that have this icon (  ) are generated transformations.

#### **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 Transformations tree.
2. Right-click the transformation and click **Analyze**. (You can also click **Analyze** in the **Actions** menu.) The Report view of the Impact Analysis window displays, as shown in the following display.

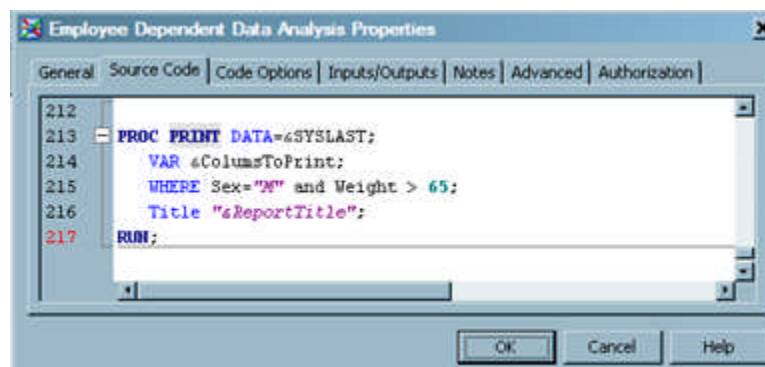
**Display 11.17** Impact Analysis on a Sample Generated Transformation

The selected generated transformation is named Employee Dependent Data. The Impact Analysis window shows that the selected transformation is used in a job. You can right-click the objects in the Report view to access their properties windows and view the jobs that contain them. For a data-flow view of the impacts, click **Diagram View**.

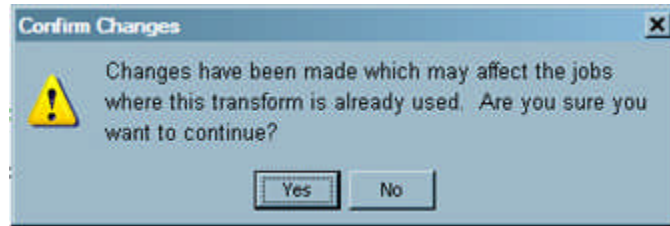
### Update Generated Transformations

Perform the following steps to update the source code and other properties of a generated transformation. Any change that you make to the generated transformation can affect existing jobs that contain the transformation.

1. Access the properties window of the transformation that you want to update by double-clicking the transformation's name in the Transformations tree.
2. Click on a tab that you want to update.
3. Make any needed changes to the source code. Click **OK** to save these changes to the SAS code. The following display depicts an update to the source code of a sample transformation.

**Display 11.18** Sample Code Tab with Updates

*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 11.19** Confirm Changes Warning

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.

---

## Editing the Generated Code for a Job or Transformation

### Problem

You want a result that cannot be easily achieved with the code that is generated for a job or transformation. Only a few changes are needed to the generated code.

### Solution

You can edit the generated code for a job or transformation and save the edited code to the metadata server or to a separate file. If you save the code to a file, 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. The specified user-written code is retrieved whenever code for this job or transformation is generated.

### 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 the job's metadata or a file:

1. Open the **Code** tab in the properties window for the job or transformation.
2. Select **User written body** or **All user written** in the **Code generation mode** field. Any portion of the code that is not user-written is dimmed when you click **User written body**. You cannot modify this part of the code.
3. Place the cursor in an editable section of the **Code** tab. Edit the generated code in the **Code** tab.
4. Click **Save** or **Save As** on the toolbar for the tab. The **Save** option enables you to save the code in the editor as a metadata object (instead of saving the code into a file). The **Save As** option opens the Save File window, where you can either save a name and description for the metadata object (code in the editor) or save the contents of the editor as a file.



*Note:* The **Save** and **Save As** options apply your changes to the current session. To make your changes persist after the current session, you must save the entire job. To save the entire job, select **File** ⇒ **Save** from the menu bar on the desktop.

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

---

## Replacing the Generated Code for a Job or Transformation

### Problem

You want a result that cannot be easily achieved with the code that is generated for a job or transformation. Extensive changes are needed to the generated code.

### Solution

You can write a SAS program to achieve the desired result. Then you can replace the generated code for the job or transformation with your program. You can copy your code into the metadata for the transformation or job (Import SAS Code), or you can specify a path to a file that contains your SAS program (Attach to SAS Code). If you change an attached source file later, the changes are reflected in the code that you update.

### Tasks

#### **Replace the Generated Code for a Job or Transformation**

Perform the following steps to replace existing code into a job or transformation.

1. Open the **Code** tab in the properties window for the job or transformation.
2. Click **User written body** or **All user written** in the **Code generation mode** field. Note that any non-user-written portion of the code is dimmed when you click **User written body**. You cannot modify this part of the code.
3. Place the cursor in an editable section of the **Code** tab.
4. Click the **Open** icon on the toolbar of the **Code** tab.
5. Click either **Import SAS Code** or **Attach to SAS Code**. Then you can copy the SAS code that is contained in the selected file into the **Code** tab of a job or transformation.

*Note:* When you click **Import SAS Code**, the code is copied without establishing a link to the source file. If you change an imported source file later, the changes are not reflected in the code that you update. However, when you click **Attach to SAS Code**, the code is copied with a link to the source file. If you change an attached source file later, the changes are reflected in the code that you update.

6. Click **Local** or **Remote** to access the Open window. The Local window enables you to open a file from your client computer. The Remote window enables you to open a file from the SAS Application Server.

*Note:* Both local and remote access are available for the import SAS code function. Only remote access is available for the attach to SAS code function.

7. Click **Save** or **Save As** on the toolbar for the tab. The **Save** option enables you to save the code in the editor as a metadata object (instead of saving the code into a file). The **Save As** option opens the Save File window, where you can either save a name and description for the metadata object (code in the editor) or save the contents of the editor as a file.

*Note:* The **Save** and **Save As** options apply your changes to the current session. To make your changes persist after the current session, you must save the entire job.

To save the entire job, select **File** ⇒ **Save** from the menu bar on the desktop.

8. Click **OK** to apply the changes to the current session and close the properties window.

## Chapter 12

# Optimizing Process Flows

---

<b>About Process Flow Optimization</b> .....	<b>231</b>
<b>Managing Process Data</b> .....	<b>232</b>
Problem .....	232
Solution .....	232
Tasks .....	232
<b>Managing Columns</b> .....	<b>235</b>
Problem .....	235
Solution .....	235
Tasks .....	236
<b>Streamlining Process Flow Components</b> .....	<b>237</b>
Problem .....	237
Solution .....	237
Tasks .....	237
<b>Using Simple Debugging Techniques</b> .....	<b>238</b>
Problem .....	238
Solution .....	239
Tasks .....	239
<b>Using SAS Logs</b> .....	<b>242</b>
Problem .....	242
Solution .....	242
Tasks .....	242
<b>Reviewing Temporary Output Tables</b> .....	<b>244</b>
Problem .....	244
Solution .....	244
Tasks .....	244
<b>Additional Performance Optimization Information</b> .....	<b>246</b>

---

## About Process Flow Optimization

Efficient process flows are critical to the success of any data management project, especially as data volumes and complexity increase. The following sections describe improving the performance of process flows in SAS Data Integration Studio with the following techniques:

- “Managing Process Data” on page 232

- [“Managing Columns” on page 235](#)
- [“Streamlining Process Flow Components” on page 237](#)

The remaining sections describe analyzing the performance of process flows that have already been created by with the following techniques:

- [“Using Simple Debugging Techniques” on page 238](#)
- [“Using SAS Logs” on page 242](#)
- [“Reviewing Temporary Output Tables” on page 244](#)

## See Also

[“Additional Performance Optimization Information” on page 246](#)

---

## 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 Views and Physical Tables” on page 232](#)
- [“Delete Intermediate Files” on page 233](#)
- [“Cleanse and Validate Data” on page 235](#)
- [“Minimize Remote Data Access” on page 235](#)

### Tasks

#### **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 is 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. When data is in 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 Job Editor window to access the **Create as 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.

In some cases you can switch the format of a permanent output table between a physical table and a view. You can right-click the permanent output table in the Job Editor window, select **Properties**, click the **Physical Storage** tab, and then select or deselect the **Create as view** option for the table. If the transformation that creates the table can create views, then the table will be created as a view. Some transformations do not support views and might ignore the setting.

### **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 Job Editor window is closed or the current server session is ended in some other way (for example, by selecting **Actions** ⇒ **Stop** from the menu. For information about how transformation output tables can be used to debug the transformations in a job, see [“Reviewing Temporary Output Tables” on page 244](#). However, as long as you keep the job open in the Job Editor 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 Job Editor 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("sqllobs"), 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 210](#).

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 deletes these output tables and free up valuable disk space and memory. For example, you can add a generated transformation that deletes

output tables at a certain point in the flow. For details about generated transformations, see [“Creating and Using a Generated Transformation” on page 219](#).

### ***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, data is considered local in a SAS Data Integration Studio job when it is directly accessible from the same machine, from a machine that is directly addressable from the primary machine, or through one of the SAS/ACCESS methods. Otherwise, it is considered remote.

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:

- [“Drop Unneeded Columns” on page 236](#)
- [“Avoid Adding Unneeded Columns” on page 236](#)
- [“Aggregate Columns for Efficiency” on page 237](#)
- [“Match the Size of Column Variables to Data Length” on page 237](#)

## 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 then build your own transformation output table columns to match your ultimate target table and map.

Finally, you can control column mapping and propagation at a job level, at a transformation level, or even at a column level. Column propagation is the ability to automatically propagate columns through the intermediate tables in a process flow to the target table. If you don't need to map or propagate some of the columns in a flow, use one of the following options:

- **Automatically map columns** and **Automatically propagate columns** options at **Tools** ⇒ **Option** ⇒ **Job Editor** (for new jobs)
- **Map Columns** and **Propagate Columns** in the pop-up menu for a job or transformation (for selected jobs and transformations)
- **Map all columns**, **Map selected columns**, **Propagate from sources to targets**, **Propagate from targets to sources**, and **Propagate columns** on the **Mappings** tab for a job or transformation (for selected jobs and transformations)

For information about mapping columns, see [“Maintaining Column Mappings” on page 154](#). For information about column propagation, see [“Managing the Scope of Column Changes in Jobs” on page 158](#).

### 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, which are done on the **Mappings** tab in the details pane of the Job Editor window or from the **Mappings** tab of the transformation, often result in the generation of an intermediate SQL view step. In many situations, that intermediate step adds processing time. In turn, these changes to columns can be propagated throughout the job. 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” on page 237](#)
- [“Use Transformations for Star Schemas and Lookups” on page 238](#)
- [“Use Surrogate Keys” on page 238](#)

**Tasks****Work From Simple to Complex**

When you build process flows, build by validating jobs as you build up complexity. For example, build a job subsection, and then test and validate it. Then, then add additional components, which you can test and validate as you go. This step-by-step process of progressively building complexity into a job is supported by the following features:

- the ability to test the validity of the subsections by using the options for **Run From Selected Transformation**, **Run To Selected Transformation**, and **Run Selected Transformations**
- the ability to test each subsection by using **Step** and **Continue** to step through and validate each subsection of the entire process

- the ability to verify the success of the job or its subsections by monitoring the **Status**, **Warnings and Errors**, and **Statistics** tabs on the Details pane of the Job Editor window
- the ability to select specific transformations for inclusion in the bar chart of performance statistics on the **Statistics** tab

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 239](#).

### ***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 [“About Slowly Changing Dimensions” on page 348](#).

### ***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, if the operational system changes its key length or type, then a surrogate key remains valid. An operational key does 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. A Surrogate Key Generator transformation 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 [“About Slowly Changing Dimensions” on page 348](#).

---

## **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 identify which steps are performing poorly. Finally, if you know what SAS options are set and how they are set, this information can help you determine what is going on in your process flows. You can perform the following tasks:

- [“Check the Status of a Job” on page 239](#)
- [“Verify Output From a Transformation” on page 239](#)
- [“Limit Input to a Transformation” on page 239](#)
- [“Add Debugging Code to a Process Flow” on page 240](#)
- [“Set SAS Invocation Options on Jobs” on page 241](#)
- [“Set and Check Status Codes” on page 241](#)

## Tasks

### **Check the Status of a Job**

You can see information about the status of your jobs and the nodes that they contain. This status information is provided by the following features:

- the status indicators and sticky note windows on the nodes on the **Diagram** tab of the Job Editor window. These features are available before and after you submit a job. Therefore, they are useful as tools that help you construct a job and determine whether it is ready to run.
- the **Status** tab on the Details pane of the Job Editor window. This feature displays the status of each node in a job as it is run. You can double-click an error or warning status on a node to display it in the **Warnings and Errors** tab.
- the **Warnings and Errors** tab on the Details pane of the Job Editor window. This feature displays any warnings or errors that are displayed as a job is run. You can click the link in an error or warning to see it displayed in the **Log** tab of the **Job Editor** window.

For information about using these features, see [“Reviewing a Successful Job” on page 145](#) and [“Diagnosing and Correcting an Unsuccessful Job” on page 150](#).

### **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 89](#).

### **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= system option for an entire job in SAS Data Integration Studio, add the following code to the **Precode and Postcode** tab in the job's property window:

```
options
obs=<number>;
```

To specify the OBS= system option 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 210](#).

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 Job Editor, this option stays in effect for all runs of the job until it is reset or the Job Editor window is closed.

The syntax for resetting the option is as follows:

```
options
obs=MAX;
```

*Note:* Removing the OBS= line of code from the Job Editor does not reset the OBS= system option. You must reset it as shown or by closing the Job Editor window.

The **Max Input Rows** option enables you to specify the number of input rows to an SQL query within the Designer window of the SQL join transformation. To access this option, click **SQL Join** in the Navigate pane of the window. Then, look for the option in the SQL Join Properties pane. You can also specify the number of output rows with the **Max Output Rows** option.

### 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 12.1** Methods for Adding Custom Debugging Code

Method	Documentation
Replace the generated code for a transformation with user-written code.	<a href="#">“Replacing the Generated Code for a Job or Transformation” on page 229</a>
Add the User-Written Code transformation to the process flow.	<a href="#">“Adding a User Written Code Transformation to a Job” on page 216</a>
Add a generated transformation to the process flow.	<a href="#">“Creating and Using a Generated Transformation” on page 219</a>
Add a return code to the process flow.	<a href="#">“Set and Check Status Codes” on page 241</a>

Custom code can direct information to the log or to alternate destinations such as external files, or 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

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 211](#).

To set SAS global options for a particular job or transformation within a job, you can add these options to the **Precode and Postcode** tab in the properties window. For more information about adding code to this window, see [“Specifying Options for Jobs” on page 210](#).

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 210](#).

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. The log can be defined to display only the transformations that affect the problem being investigated, making the log more manageable and eliminating inconsequential error messages. For more information about status code handling for transformations, see [“Perform Actions Based on the Status of a Transformation” on page 175](#).

You should also remember that the status code information is supplemented by the job and node status information in the Job Editor window, particularly the **Status** tab and **Warnings and Errors** tab in the Details pane. For more information, see [“Check the Status of a Job” on page 239](#).

---

## 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” on page 242
- “Capture Additional SAS Options in the SAS Log” on page 242
- “View or Hide SAS Logs” on page 243
- “Redirect Large SAS Logs to a File” on page 243

### **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 a hardware issue, see the document “A Practical Approach to Solving Performance Problems with the SAS System,” which 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.

If you need to examine additional performance statistics, you can right-click in an open job and click **Collect Runtime Statistics** in the pop-up menu. After you run the job, you can review the statistics that are generated in the run on the **Statistics** tab of the Details pane. You can display the statistics in the form of a table, a line graph, or a bar chart.

#### ***Capture Additional SAS Options in the SAS Log***

Another way to analyze performance is to turn on the following SAS options so that detailed information about the SAS tasks 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 the document "A Practical Approach to Solving Performance Problems with the SAS System," which is available from the "Scalability and Performance Papers" page at <http://support.sas.com/rnd/scalability/papers/>.

In addition, the following SAS statements 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 sends 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 sends information (such as physical path location and the engine that is 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 241.

### 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 Job Editor 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 Job Editor 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 contain the name of the job that creates the log, plus a timestamp of when the job is executed.

Alternatively, you can add the following code to the **Precode and Postcode** 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 210](#). This code causes the log to be redirected to the specified file. Be sure to use the appropriate host-specific syntax of the host where your job is running when you specify this log file, and make sure that you have Write access to the location where the log is 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 view a transformation's temporary output table from the Process Designer window and 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” on page 244](#)
- [“View Temporary Output Tables” on page 245](#)
- [“Redirect Temporary Output Tables” on page 245](#)
- [“Add the List Data Transformation to a Process Flow” on page 246](#)
- [“Add a User-Written Code Transformation to the Process Flow ” on page 246](#)

### 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 241](#).

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
9.2\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 Job Editor 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 click **Open**. The transformation's output table is displayed in the View Data window.

This approach works if you do not close the Job Editor window. When you close the Job Editor window, the current server session ends, and the output tables are deleted. For information, see [“Browsing Table Data” on page 89](#).

### **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 that are 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 Job Editor 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\9.2";
proc copy move
in = work
out = permlib;
select test1 test2 test3;
run;
```

For information about User Written Code transformations, see [“Adding a User Written Code Transformation to a Job”](#) on page 216.

---

## Additional Performance Optimization 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 322. For specific information about the performance of the Table Loader transformation, see [“Selecting a Load Technique”](#) on page 273 and [“Removing Non-Essential Indexes and Constraints during a Load”](#) on page 276.

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/resources/papers/>.

## Chapter 13

# Using Impact Analysis

---

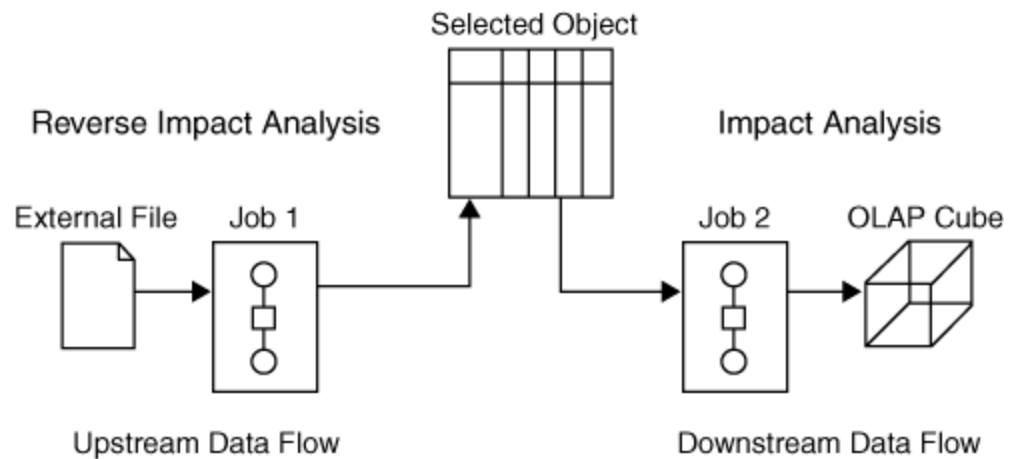
<b>About Impact Analysis and Reverse Impact Analysis</b> . . . . .	<b>247</b>
<b>Performing an Impact Analysis</b> . . . . .	<b>248</b>
Problem . . . . .	248
Solution . . . . .	249
Tasks . . . . .	249
<b>Performing Impact Analysis on a Generated Transformation</b> . . . . .	<b>251</b>
Problem . . . . .	251
Solution . . . . .	252
Tasks . . . . .	252
<b>Performing Reverse Impact Analysis</b> . . . . .	<b>253</b>
Problem . . . . .	253
Solution . . . . .	253
Tasks . . . . .	253

---

## About Impact Analysis and Reverse Impact Analysis

Impact analysis identifies the tables, columns, jobs, and transformations that are affected by a change to 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. Use impact analysis before changing or deleting a metadata object, to see how that change can affect other objects. Use reverse impact analysis to trace the source data that contributes to the content of a selected table or column.

The following figure shows the difference between impact analysis and reverse impact analysis for a selected object.

**Figure 13.1** 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.

Analysis is performed on all metadata repositories on the current metadata server. You can generate impact and reverse impact analyses for most types of data objects, including columns, tables, external files, information maps, Enterprise Guide projects and associated objects, and the levels and measures in OLAP cubes. You can also generate impact analyses for generated transformations, as described in [“Performing Impact Analysis on a Generated Transformation”](#) on page 251.

To perform an analysis, right-click an object in the Inventory tree, Custom tree, or Job Editor and select **Analyze**. This action opens a new window that contains up to four tabs, which include Impact Analysis, Reverse Impact Analysis, Contents, and Reports. Analytical results appear in the Impact Analysis or Reverse Impact Analysis tabs. In those tabs, you can right-click on the table and select **Analyze Columns** to determine how that table or job impacts or is impacted by the selected object. Within these tabs, you can also display properties or select **Open** to view the data in a table. You can also select one of the icons at the top of the tab to view the object in a tree or diagram view or to print the contents.

If you run an analysis and the results do not include objects that you know exist on the system, ask your administrator to verify that you have the appropriate privileges to see these objects. For more information, the administrator should see the *SAS Intelligence Platform: Security Administration Guide*.

---

## Performing an 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 would 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.

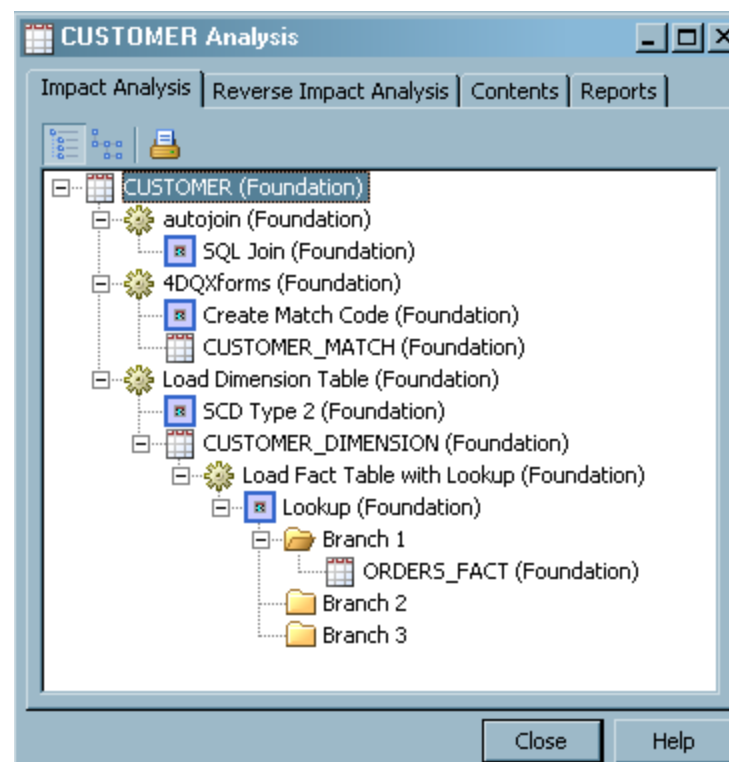
## Tasks

### Perform an Impact Analysis

To perform impact analysis on a metadata object, right-click the object in a tree view or in a process flow in the Job Editor window, and then select **Analyze** from the pop-up menu. Be sure to save the job in the **Job Editor** window before running analysis on a metadata object in that job. Otherwise, your analysis does not reflect any changes since the last save.

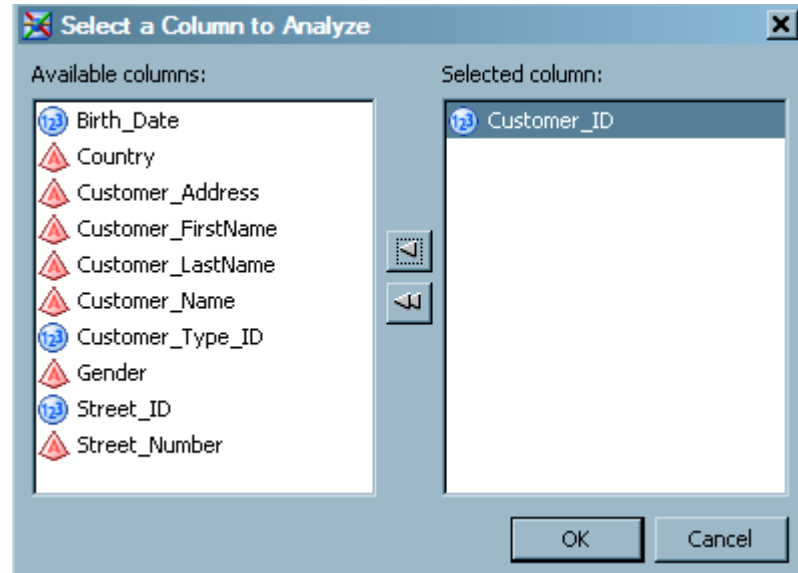
Alternatively, you can select the object in a tree view or in the context of a process flow, select **Actions** from the menu bar, and then select **Analyze**. The following display shows the tree view of the analysis of a table named CUSTOMER.

**Display 13.1** Impact Analysis Tab

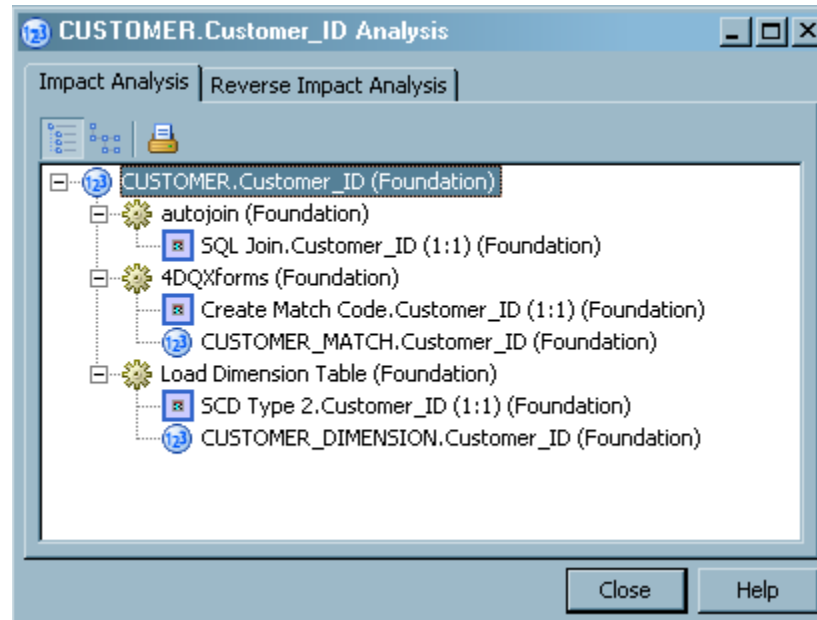


Perform the following steps to trace the impact of the metadata for a table column:

1. In a tree view or in the context of a process flow, right-click on the metadata object for the table that contains the column to be analyzed. Select **Analyze**.
2. In the Analyze window, right-click on the metadata object for the table, then select **Analyze Columns**.
3. Select the column you want from the **Available columns** pane. Use the arrow key to move it to the **Selected column** pane.

**Display 13.2** Select a Column to Analyze Window

4. Click the **OK** button. A new window appears. In the following display, this window shows the result of an analysis performed on a column named Customer\_ID in a table named CUSTOMER.

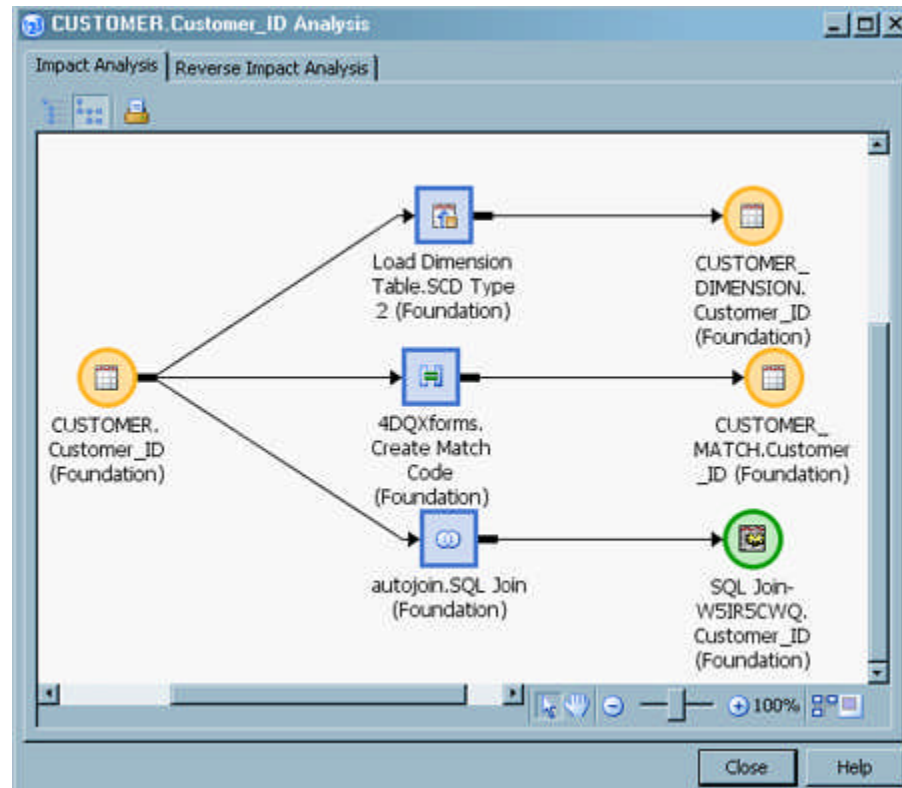
**Display 13.3** Analysis Results

The **Tree View** window 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 three jobs. In this example, the third job 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.
- **Load Dimension Table (Foundation)**: specifies the job, Load Dimension Table, to which the Customer\_ID column is an input. The mapping type is 1:1.

- **SCD Type 2 Loader.Customer\_ID (1:1) (Foundation)**: specifies the transformation that 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.
5. To view the results as a graphical display, click on the icon for the Diagram View. The same analytical results as shown in the preceding hierarchical display are shown in the following graphical example.

**Display 13.4** Analysis Diagram View



The Diagram View 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 on a metadata server. After they are registered, your transformations

display in the Transformations tree, where they are available for use in any job. For more information about these transformations, see [“Creating and Using a Generated Transformation” on page 219](#).

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 might be affected by the change.

## Solution

Run impact analysis on a generated transformation.

## Tasks

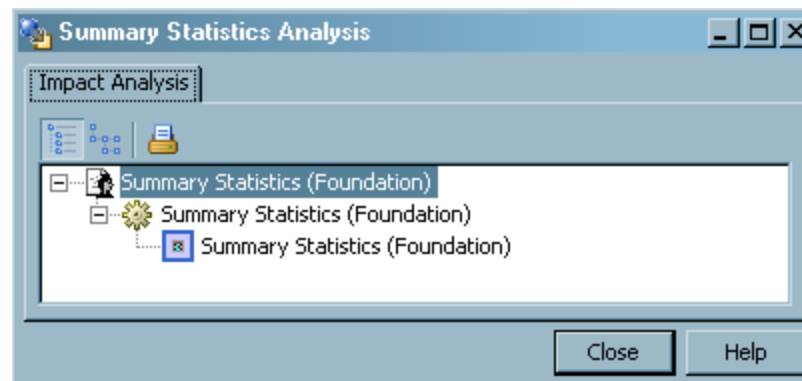
### Perform Impact Analysis on a Generated Transformation

Perform the following steps to run an impact analysis on a generated transformation:

1. From the SAS Data Integration Studio desktop, select the Transformations or Inventory tree.
2. Open the folder that contains the generated transformation that you want to analyze.
3. Select that transformation, right-click the object, and select **Analyze** 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 **Actions** from the menu bar, and then select **Analyze**. The following display shows the tree view of the analysis.

**Display 13.5** Impact Analysis on a Generated Transformation



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 Summary Statistics.

You can right-click the objects on the **Impact Analysis** tab to obtain information about those objects.

For a process flow view of the impacts, select the **Diagram view** icon.



---

## 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***

To perform impact analysis on a metadata object, right-click the object in a tree view or in a process flow in the Job Editor window, and then select **Analyze** from the pop-up menu. Be sure to save the job in the Job Editor window before running analysis on a metadata object in that job. Otherwise, your analysis does not reflect any changes since the last save.

Alternatively, you can select the object in a tree view or in the context of a process flow, select **Actions** from the menu bar, and then select **Analyze**.

Once the Analysis window opens, select the **Reverse Impact Analysis** tab. The steps for performing reverse impact analysis on a column are similar to the steps in [“Perform an Impact Analysis” on page 249](#).



## Chapter 14

# Working with Reports

---

<b>About Reports</b> .....	<b>256</b>
<b>Opening the Reports Window</b> .....	<b>256</b>
Problem .....	256
Solution .....	256
Tasks .....	256
<b>Selecting the Reports Perspective</b> .....	<b>257</b>
Problem .....	257
Solution .....	257
Tasks .....	258
<b>Customizing the Tables Report</b> .....	<b>258</b>
Problem .....	258
Solution .....	258
Tasks .....	259
<b>Customizing the Job Documentation Report</b> .....	<b>259</b>
Problem .....	259
Solution .....	259
Tasks .....	260
<b>Running and Saving a Report</b> .....	<b>260</b>
Problem .....	260
Solution .....	260
Tasks .....	261
<b>Saving a Report As a Document Object</b> .....	<b>262</b>
Problem .....	262
Solution .....	262
Tasks .....	262
<b>Viewing a Report</b> .....	<b>263</b>
Opening a Report .....	263
Contents of a Tables Report .....	263
Contents of a Job Report .....	264
Contents of Your Own Report .....	265
<b>Creating Your Own Report</b> .....	<b>265</b>
Problem .....	265
Solution .....	265
Tasks .....	265

## About Reports

The reports feature in SAS Data Integration Studio can be used to generate reports. You can generate reports to review the metadata for tables and jobs in a convenient format. You can generate your own reports by creating a Java report plug-in. For more information about generating your own reports see [“Creating Your Own Report” on page 265](#).

Reports enable you to:

- find information about a table or job quickly
- compare information between different tables or jobs
- obtain a single file that contains summary information of all tables or jobs in HTML, RTF, or PDF format
- perform custom behaviors that are defined by user-created plug-in SAS code, Java code, or both

You can access reports in SAS Data Integration Studio using document objects. You can save the physical path to a report as a document object, and access that document object in the Folders tree or the Inventory tree on the SAS Data Integration Studio desktop. For more information about accessing reports with document objects see [“Saving a Report As a Document Object” on page 262](#).

---

## Opening the Reports Window

### **Problem**

You want to view the Reports window.

### **Solution**

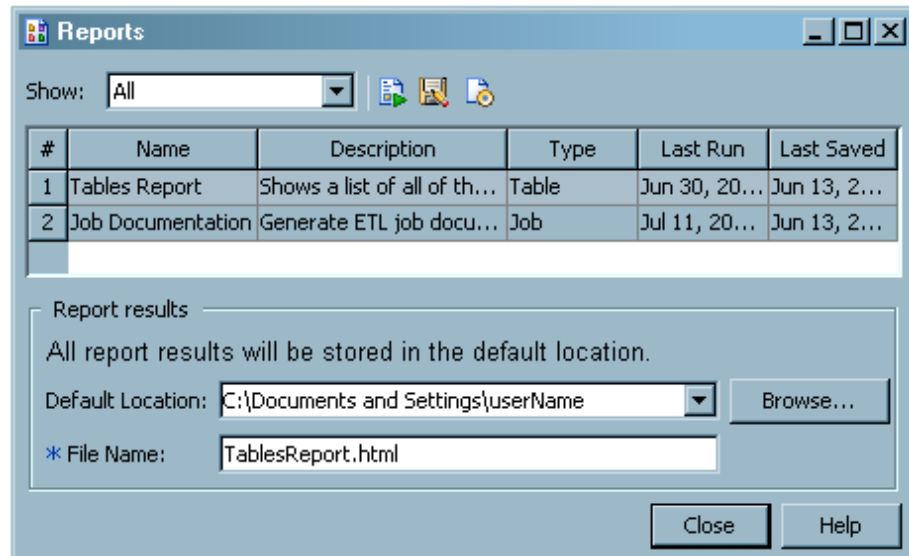
You can view the Reports window by using the drop-down menu in the **Tools** field or the **Reports** button on the SAS Data Integration Studio menu bar.

### **Tasks**

#### ***Access the Reports Window***

Perform the following steps to access the Reports window.

1. Select **Tools** on the SAS Data Integration Studio menu bar.
2. Click **Reports** on the drop-down menu in the **Tools** field, or you can click the **Reports** button on the SAS Data Integration Studio menu bar to open the Reports window.

**Display 14.1** Reports Window

The Reports window contains the following information about a report:

- the name of a report
- a description of a report
- the type of report
- the time the report was last run
- the time the report was last saved

You can sort multiple reports that are listed in the Reports window by their number. You can also sort reports alphabetically by their name, description, type, date last run, or date last saved. For example, clicking once on the **Name** tab sorts all reports in the Reports window in increasing alphanumeric order, and an arrow pointing up appears on the **Name** tab. Clicking a second time on the **Name** tab sorts all reports in the Reports window in decreasing alphanumeric order, and an arrow pointing down appears on the **Name** tab.

---

## Selecting the Reports Perspective

### Problem

You want to choose a perspective in the Reports window that includes only reports about tables, jobs, or any additional report plug-in categories.

### Solution

You can use the drop-down menu in the **Show** field in the Reports window to choose a perspective that includes all reports or just reports about tables, jobs, or any additional categories.

## Tasks

### Select the Perspective that Includes Tables, Jobs, or all Categories

Perform the following steps to select the perspective that includes tables, jobs, or all categories.

1. Open the Reports window in SAS Data Integration Studio.
2. Click the drop-down menu in the **Show** field at the top of the Reports window. The following table lists the possible options in the drop-down menu in the **Show** field and describes their effect on the perspective in the Reports window. The drop-down menu in the **Show** field displays any additional report plug-in categories after the categories of Table and Job, and before the category Recently Run.

**Table 14.1** Perspective Options on the Show Drop-down Menu

Option	Description
All	Selects a perspective that shows all reports.
Table	Selects a perspective that includes all reports in the Table category.
Job	Selects a perspective that includes all reports in the Job category.
Recently Run	Shows a perspective that includes all reports that have a date in the Last Run column in the table.
Saved As Documents	Shows only the reports that have a date in the Last Saved column in the table.

---

## Customizing the Tables Report

### Problem

You want to customize the generated Tables Report.

### Solution

You can specify the format type, style sheet, and additional Output Delivery System (ODS) options to modify how the Tables Report is generated and control where the report output is saved.

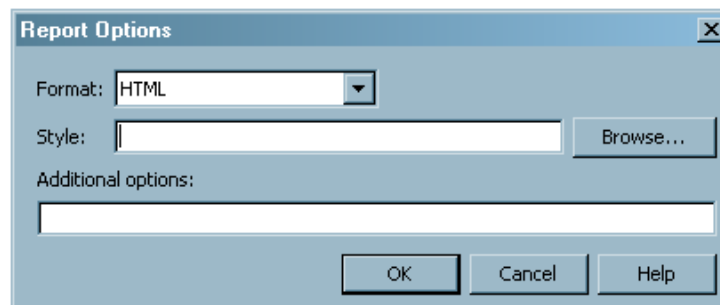
## Tasks

### Specify Format and Style Changes for a Tables Report

Perform the following steps to specify format and style changes for a Tables Report.

1. Open the Reports window in SAS Data Integration Studio.
2. Click on the Tables Report so that it is highlighted. If you do not see Tables Report make sure the perspective is set to **Table** or **All** in the drop-down menu in the **Show** field.
3. Click **Additional report options** at the top of the Reports window. After you click the **Additional report options** button, the following ODS Report Options dialog box is shown.

**Display 14.2** Report Options Dialog Box for Tables and Plug-in Code



4. Click the drop-down menu in the **Format** field to format your report as an HTML, RTF, or PDF file.
5. (Optional) Specify a path to a style for your report in the **Style** field, or click **Browse** to search for a path. For more information about style sheets, see the *SAS Output Delivery System User's Guide*.
6. (Optional) Specify additional Output Delivery System (ODS) options in the **Additional options** field. For more information about ODS options, see the *SAS Output Delivery System User's Guide*.
7. Click **OK** to save your ODS report options, or click **Cancel** to keep the default report options.

---

## Customizing the Job Documentation Report

### Problem

You want to customize the generated Job Documentation Report.

### Solution

You can specify how to customize the generated Job Documentation Report with the **Additional report options** button and the Report results pane in the Reports window.

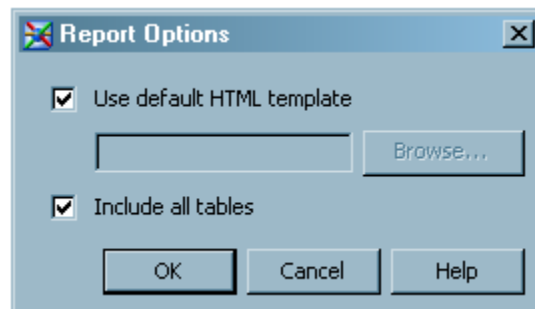
## Tasks

### Specify Job Report Options

Perform the following steps to specify job report options.

1. Open the Reports window in SAS Data Integration Studio.
2. Click on the Job Documentation Report so that it is highlighted. If you do not see a job report, make sure the perspective is set to **All** or **Job** in the drop-down menu in the Show field.
3. Click **Additional report options** at the top of the Reports window. After you click the **Additional report options** button, the following Job Documentation Report Options dialog box opens.

**Display 14.3** Job Documentation Report Options Dialog Box



The default settings for a job documentation report use the default HTML page, index.html, and include all tables. To specify a different template for your job documentation report, deselect the **Use default HTML template** check box, and enter the path to another template in the text box. Alternatively, click **Browse** to search for a template. Deselect the **Include all tables** check box to include only those tables that have been registered in the Folders tree on the SAS Data Integration Studio desktop.

4. Click **OK** to save your job documentation report options, or click **Cancel** to keep the default job documentation report options.

---

## Running and Saving a Report

### Problem

You want to run and save a report.

### Solution

You can run and save a report by using the **Run and view a report** button on the Reports window.



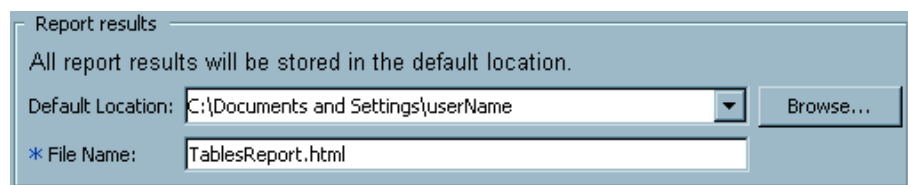
## Tasks

### Run and Save a Report

Perform the following steps to run and save a report.

1. Open the Reports window in SAS Data Integration Studio.
2. Click on a report in the Reports window so that it is highlighted. If you do not see the report you want, verify that the perspective in the Reports window includes the type of report you want by checking the drop-down menu in the **Show** field.
3. Edit your report's name in the **File Name** field in the Report results pane of the Reports window.

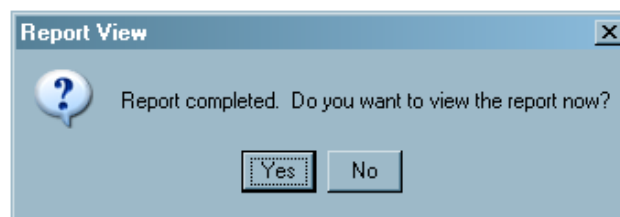
**Display 14.4** Report Results Pane



4. Check the default location to save your report in the **Default Location** field in the Report results pane. This location is on the default SAS Application Server for SAS Data Integration Studio, which is probably not the computer where SAS Data Integration Studio is installed. You can change the directory to save your report by entering a new path in the **Default Location** field. Alternatively, click **Browse** to navigate to the directory of your choice. It is a good idea to use the **Browse** button to examine the file folder hierarchy and check the path.
5. Click **Run and view a report** at the top of the Reports window. Alternatively, you can double-click on a report in the Reports window to run and save a report.

Your report is saved to the path specified in the **Default Location** field in the Report results pane of the Reports window. After you click the **Run and view a report** button, or double-click a report, a Report View dialog box will open once the report has been successfully created. A plug-in report might be designed to behave differently.

**Display 14.5** Report View Dialog Box



6. Click **Yes** to view the report, or click **No** to close the Report View dialog box. Note that a report opens only if the **Default Location** field in the Report results pane contains a valid path. A plug-in report might be designed to behave differently. For more information about viewing a report see [“Viewing a Report” on page 263](#).

## Saving a Report As a Document Object

### Problem

You want to save a report as a document object, so that you can access this report from the SAS Data Integration Studio Folders tree.

### Solution

You can save a report as a document object by using the **Save the report result as a document object** button on the Reports window.

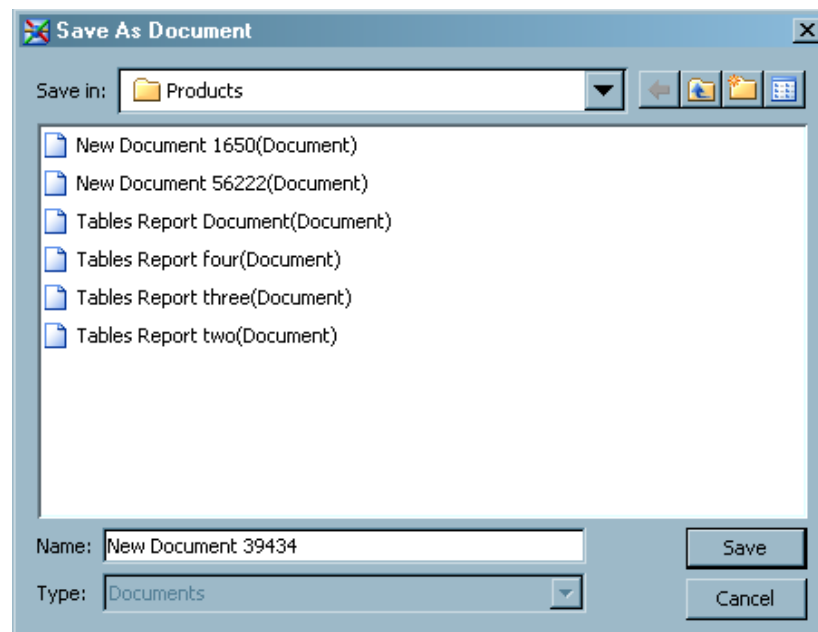
### Tasks

#### Save a Report As a Document Object

Perform the following steps to save a report as a document object.

1. Open the Reports window in SAS Data Integration Studio.
2. Click on a category in the Reports window so that it is highlighted. If you do not see the report you want, verify that the perspective in the Reports window includes the type of report you want by checking the drop-down menu in the **Show** field.
3. Click **Save the report as a document object** at the top of the Reports window. After you click the **Save the report as a document object** button, a Save As Document dialog box will open. You can use the drop-down menu in the **Save in** field to specify the location in the Folders tree on the SAS Data Integration Studio desktop to save your document object. Choose a name in the **Name** field for your document object.

**Display 14.6** Save As Document Dialog Box



- Click **Save** to create your document object, or **Cancel** to close the Save As Document dialog box.

*Note:* A document object will not open a report if the report is moved to a different directory. This is because a document object contains the path where the HTML file was originally created.

## Viewing a Report

### Opening a Report

You can open a report one of the following ways.

- Click **Yes** on the Report View dialog box after clicking **Run and view a report** on the Reports window.
- Right-click a document object in the Folders tree on the SAS Data Integration Studio desktop, and select **Open**.
- Navigate to the directory on your computer or network where the report is saved and double-click on the report icon.

### Contents of a Tables Report

A tables report contains information about the tables in the Inventory tree on the SAS Data Integration Studio desktop. See the following display for a portion of a sample tables report.

**Display 14.7** Tables Report

Tables Report				
Obs	Table Name	Description	Created	Last Modified
1	ADVERSE		11Jan2008:20:24:22	02Apr2008:17:52:29
2	ADVERSE_SORTED		11Jan2008:20:24:24	11Jan2008:20:24:24
3	ALL_EMP		11Jan2008:20:24:27	02Apr2008:15:31:13
4	ALL_EMP2	ALL_EMP2	11Jan2008:20:24:29	11Jan2008:20:24:29
5	AREA	Area	11Jan2008:20:24:32	11Jan2008:20:24:32
6	BANKACCOUNTS	Bank accounts	11Jan2008:20:24:34	11Jan2008:20:24:34

A tables report contains:

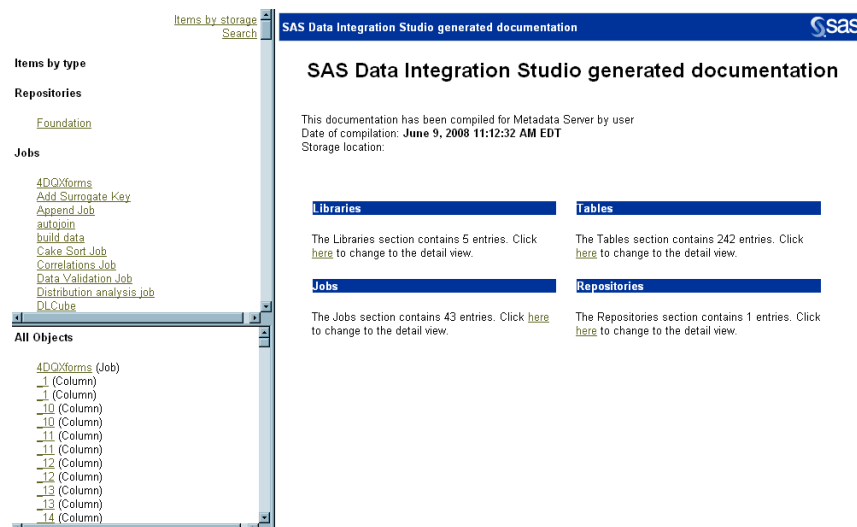
- an observation number for each table
- the name of a table
- a description of the table
- the date that the table was created
- the date that the table was last modified
- the owner of the table
- the schema of the table

- the folder where the table resides in the Folders tree on the SAS Data Integration Studio desktop
- the date that the table was checked out

## Contents of a Job Report

A job report contains three windows. The first window is the Main window for the job report, and is located on the right. The second window is an Items window, and it is located in the upper left hand corner of the job report. The third window is an Objects window, and it is located in the lower left hand corner of the job report.

### Display 14.8 Job Report



### Main Window

The Main window contains links to detailed information about libraries, tables, jobs, and metadata repositories.

### Items Window

An item is a metadata repository, job, library, or table.

The Items window allows you to select items by type, select items by storage, or search for an item by name.

To select an item by type, make sure the “items by type” perspective is selected in the Items window. The “items by type” perspective contains a link for each metadata repository, job, library, and table. You can open detailed information about an item in the Main window of a job report by clicking on a link for an item.

To select an item by storage, make sure the “items by storage” perspective is selected in the Items window. The “items by storage” perspective allows you to browse items in a tree as they are stored in the Folders tree on the SAS Data Integration Studio desktop. You can open detailed information about an item in the Main window of a job report by clicking on a link for an item.

To search for an item by name, make sure the “search” perspective is selected in the Items window. The “search” perspective allows you to search for an item by entering the name of the item in a text box. You can open detailed information about an item in the Main window of a job report by clicking on a link for an item that is in the results set of a search.

## Objects Window

An object is a table name or column name in a table.

The Objects window contains an alphabetical list of links for each table and column name. The Objects window is useful to look up metadata for a table if you know the name of a column in a table, but do not know the name of the table. You can open detailed information about an object in the Main window of a job report by clicking on a link for an object.

## Contents of Your Own Report

You can create your own report by writing a Java report plug-in. The content of the report can be generated by using SAS code, Java code, or both. For more information about creating your own report see [“Creating Your Own Report” on page 265](#).

---

# Creating Your Own Report

## Problem

You want to create a custom report in SAS Data Integration Studio.

## Solution

You can create a custom report by using SAS Data Integration Studio software's plug-in functionality. The Java plug-in report can generate the content of the report by using SAS code, Java code, or both.

## Tasks

### Create a Report Category

Perform the following steps to add your own report category to the Reports window. Note that these steps create the Tables Report, which you can find in the table in the Reports window.

1. Create a new Java package for:

`com.sas.reports`

that contains the file:

`TableListingReport.java`

The `TableListingReport` class extends an abstract class called `AbstractReport`. `AbstractReport` contains the implementation of the reporting plug-in interface called `ReportingInterface`. `TableListingReport` shows an implementation of only the mandatory methods that have not been implemented in `AbstractReport`. It is recommended that when creating a custom report to extend `AbstractReport` class. For an example of the `TableListingReport`, see [“Example Java Code for a Report Plug-in” on page 427](#). For explanations of the methods in the report plug-in interface, see [“Reporting Interface Methods” on page 433](#).

2. Compile `TableListingReport.java` to create class files.

3. Create a manifest file, called MANIFEST.MF, that describes your compiled classes, and add the following line to the MANIFEST.MF file:

Plugin-Init: com.sas.reports.TableListingReport.class

If you do not add this line to MANIFEST.MF, then SAS Data Integration Studio software cannot recognize this plug-in.

4. Build a compressed JAR (Java ARchives) file (not an "executable" JAR file) that contains your compiled class files, and the MANIFEST.MF file. Before adding the manifest file to the JAR file, create a folder called META-INF, and put your manifest file in this folder. Now add the META-INF folder to your JAR file.
5. Navigate to the folder called 'plugins' in the 'SASDataIntegrationStudio' folder. If SAS Data Integration Studio is installed in your Program Files, a likely path for the 'plugins' folder is:

C:\Program Files\SAS\SASDataIntegrationStudio\4.2\plugins

Once inside the plugins directory, create a new folder. You do not need to name the folder anything in particular. Add your JAR file into the folder that you just created. SAS Data Integration Studio software cannot find your JAR file if you just add it to the plugins directory, or if your JAR file is two or more directories deep from the plugins folder. You must put your JAR file inside a folder that you create in the plugins directory. If the name of the folder you created is 'reports', and the name of your JAR file is 'sas.reports.jar', then the complete path of this JAR file based on the previous example path, would be:

C:\Program Files\SAS\SASDataIntegrationStudio\4.2\plugins\  
reports\sas.reports.jar

6. Start SAS Data Integration Studio to populate the Reports window with the category that corresponds to your plug-in code in the JAR file that you created. If you do not see a report for your plug-in code in the Reports window, make sure the perspective in the Reports window is set to **All** in the drop-down menu in the **Show** field.

You can add multiple reports to your package. If you want to add multiple reports, compile class files for each report category that you want to create, and add the compiled classes to your JAR file. Modify the Plugin-Init line of code in your manifest file by adding each class, and separating each class by a semi-colon.

## Part 3

---

# Working with Transformations

<i>Chapter 15</i>	
<b>Working with Loader Transformations</b> .....	269
<i>Chapter 16</i>	
<b>Working with SAS Sort Transformations</b> .....	279
<i>Chapter 17</i>	
<b>Working with SQL Join Transformations</b> .....	285
<i>Chapter 18</i>	
<b>Working with Iterative Jobs and Parallel Processing</b> .....	333
<i>Chapter 19</i>	
<b>Working with Slowly Changing Dimensions</b> .....	347
<i>Chapter 20</i>	
<b>Working with Change Data Capture</b> .....	369
<i>Chapter 21</i>	
<b>Working with Message Queues</b> .....	379
<i>Chapter 22</i>	
<b>Working with SPD Server Cluster Tables</b> .....	389





## Chapter 15

# Working with Loader Transformations

---

<b>About Loader Transformations</b> . . . . .	<b>269</b>
<b>About the SPD Server Table Loader Transformation</b> . . . . .	<b>270</b>
<b>About the Table Loader Transformation</b> . . . . .	<b>270</b>
<b>Setting Table Loader Transformation Options</b> . . . . .	<b>271</b>
Problem . . . . .	271
Solution . . . . .	271
Tasks . . . . .	272
<b>Selecting a Load Technique</b> . . . . .	<b>273</b>
Problem . . . . .	273
Solution . . . . .	274
Tasks . . . . .	274
<b>Removing Non-Essential Indexes and Constraints during a Load</b> . . . . .	<b>276</b>
Problem . . . . .	276
Solution . . . . .	276
<b>Considering a Bulk Load</b> . . . . .	<b>277</b>
Problem . . . . .	277
Solution . . . . .	277
Tasks . . . . .	277

---

## About Loader Transformations

SAS Data Integration Studio provides three specific transformations to load data. Although most data-related transformations load data into temporary SAS WORK tables, these Loader Transformations are designed to output to permanent, registered tables (that is, tables that are available in the Folder or Inventory Tree). Loaders can create and replace tables and maintain indexes, as do the other transformations. Loaders can also do updates, appends, and be used to maintain constraints.

SAS Data Integration Studio provides the following transformations for loading data into permanent output tables:

- 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 [“Transformations That Support Slowly Changing Dimensions”](#) on page 349.

- The SPD Server Table Loader transformation 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. For more information, see [“About the SPD Server Table Loader Transformation” on page 270](#).
- The Table Loader transformation is a general loader that reads a source table and writes to a target table. This transformation can be used to load SAS and most DBMS tables, as well as Excel spreadsheets. The code generated by this transformation includes syntax that is specific to the output data type. For more information, see [“About the Table Loader Transformation” on page 270](#).

---

## About the SPD Server Table Loader Transformation

The SPD Server Table Loader transformation can be added to a process flow when a SAS Scalable Performance Data (SPD) Server table is used as a target. 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. This field is found in the Loader window of 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

You can always let a SAS Data Integration Studio transformation perform a simple load of its output table that drops and replaces the table. However, you can also add a Table Loader transformation to a permanent output table. Then, you can use the options in the Table Load transformation to control how data is loaded into the target table. In fact, a separate Table Loader transformation might be desirable under the following conditions:

- loading a DBMS table with any technique other than drop and replace
- loading tables that contain rows that must be updated upon load (instead of dropping and recreating the table each time the job is executed)
- creating primary keys, foreign keys, or column constraints
- performing operations on constraints before or after the loading of the output table
- performing operations on indexes other than after the loading of the output table

The Table Loader transformation generates code that reads a single source table (or view) and updates, replaces, or appends it to a permanent target table. Supported target types include SAS, Excel, and a wide variety of DBMS types. For data types that support constraints such as not-null and primary, unique, and foreign keys, a Table Loader can be set to generate the appropriate code to add or remove constraints. Constraint actions can be set independently for before and after the load. Likewise, the adding and removing of indexes can be controlled in the same way.

Choosing the Load Style and Technique is critical to getting the Table Loader to perform the correct task for the job efficiently. User requirements control which style (Update, Replace, or Append) to select. Once the style has been selected, a number of possible techniques to accomplish the task are presented. Choosing the correct technique is often a matter of deciding which technique will likely result in the best performance for the job when it later runs in production. The exact number and types of available styles and techniques depend on the target's data type. Some data types support clearing old rows by using a technique known as Truncate, while others do not. Some data types support a special Upsert technique, which updates rows that match on a specific key and appends the other rows to the master. Some support direct access; for those, the DATA step Modify technique is a choice. For more information about all the available techniques, see the Help topic for the Load Technique.

Once the technique is chosen, additional options that are associated with the selected technique should be reviewed to determine whether any option values should be changed from their defaults. Also, with performance in mind, you should consider any special handling of constraints and indexes.

It is important to know that non-loader transformations can load data directly into a permanent table if it has no constraints, in effect doing a **Replace Entire table** without using a Table Loader. This is done in the **Job Editor** by replacing the non-loader's output WORK table with a registered table. This technique is not supported by all transformations for all data types.

A new **Replace Simulating truncate** load style has been added for SAS targets. This choice empties the output table by using a DATA step with SET and STOP statements. This actually recreates the target table with no rows before data from the source is appended. Original data is physically deleted, not just logically deleted as with **Replace All rows using delete**. Constraints are restored as they were on the physical table before the load.

**CAUTION:**

**When using this load style, the new table structure is derived from the physical table (assuming it pre-existed) and not from metadata. This load style does not reflect changes to the column, index, or constraint metadata after the creation of the table.**

One feature that is available for SAS tables with **Replace Simulating truncate**, but not available with other Replace types, is the ability to use generation data sets. Generation data sets are a way of automatically saving a specified number of backups of the target. In SAS, this feature is enabled by adding the data set option GENMAX=#.

---

## Setting Table Loader Transformation Options

### **Problem**

You want to specify the options that control how the Table Loader transformation updates the target.

### **Solution**

You can use the settings on the **Load Technique** tab in the properties window for the Table Loader transformation. Some of the settings on the tab vary depending on which load styles that you use, although some settings appear for more than one load style.

In addition to the options on the **Load Technique** tab, more options are located under the **Options** tab in the properties window.

## Tasks

### Setting the Table Loader Job Options

Perform the following steps to set the response:

1. Create a job in SAS Data Integration Studio and give it an appropriate name.
2. Drop the Table Loader transformation from the **Process** tab onto the Job Editor window. Drag and drop a source table and a target table from the **Inventory** or **Folders** tab to the appropriate sides of the Table Loader transformation. Connect the source and target tables to the transformation. This step creates a single process flow diagram for the job, which is shown in the following example.

**Display 15.1** Sample of the Table Loader Flow



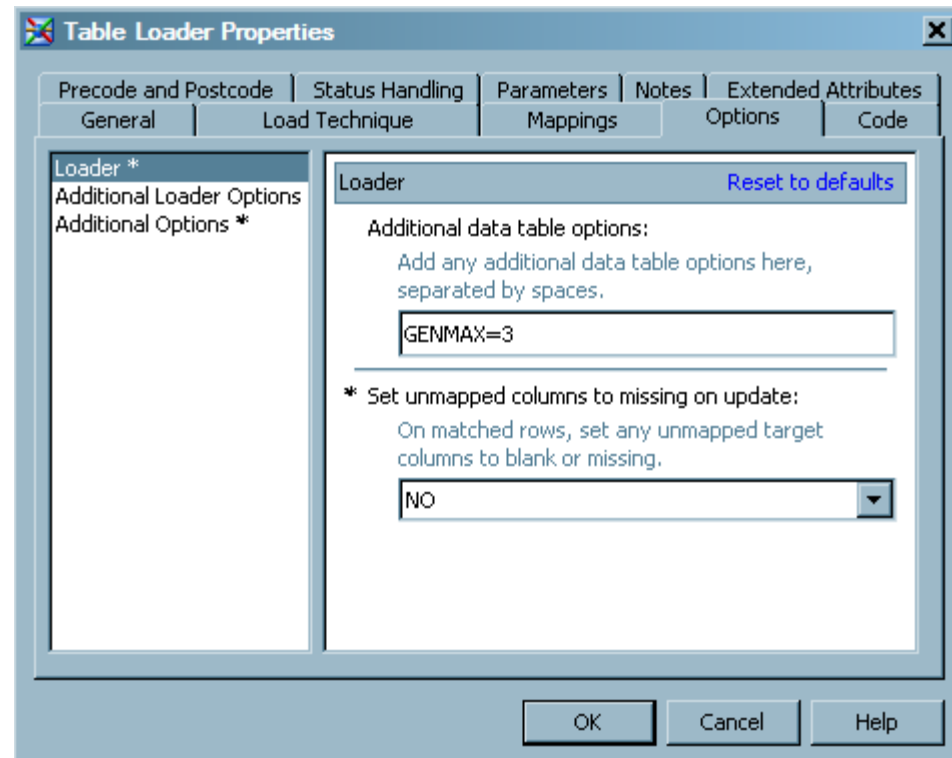
3. Set the Load Technique by right-clicking on the Table Loader transformation to open the **Properties** window. Select the **Load Technique** tab. Here you can set the load style, the technique to be used, and the constraints or indexes. For this example, which uses a SAS table, the selections are shown in the following display.

**Display 15.2** Sample Table Loader Load Technique Selections

4. If these options are not already set in the target table object, you can set additional options by selecting the **Options** tab in the Properties window. For example, your business requires that three generations of target table backups be kept, and you need

to use the load style of **Replace** with a load technique of **Simulate truncate**. Open the **Options** tab and enter `GENMAX=3` in the **Additional table options** field of the Loader window.

**Display 15.3** Modify Table Loader Options



5. Click **OK** to save the setting and close the properties window.
6. Submit and run the job.
7. Save the job.

## 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 which one of these load styles best meets your process requirements:

- appending all of the source data to any previously loaded data
- replacing all previously loaded data with the source data
- using the source data to update and add to the previously loaded data that is based on specific key columns

Once you know which load style is required, you can select the techniques and options that 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 to load SAS data sets, database tables, and other types of data, such as an Excel spreadsheet. When you load a table type that supports indexing or constraints, you can use the Table Loader transformation to manage indexes and constraints on the table.

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 can vary. The Table Loader transformation generates code to perform a combination of the following loading tasks:

- “Remove All Rows” on page 274
- “Add New Rows” on page 275
- “Match and Update Rows” on page 275

The following sections describe the SAS code alternatives for each load task and provide tips for selecting the load technique (or techniques) that performs 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 DBMS tables that support truncation)
- **Replace All rows using delete:** uses PROC SQL with DELETE \* to remove all rows
- **Replace Simulating truncate:** uses the DATA step with SET and STOP statements to remove all rows (available only for SAS tables)

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 replace all rows selections enables you to keep all indexes and constraints intact during the load. By design, using TRUNCATE is the quickest way to remove all rows. In **Replace All rows using delete**, the DELETE \* syntax also removes all rows; however, based on the database and table settings, this choice can incur overhead that can 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, the table's physical size continues to increase, which can negatively affect performance.

**Replace Simulating truncate** is available for SAS tables. It does not remove rows from a table as **Replace All rows using delete** does, or as **Replace All rows using truncate** does for a DBMS. It actually behaves more like **Replace Entire table** in that the entire table is replaced with an empty table before being loaded. Unlike **Replace All rows using delete**, this replace style does not have the issue of ever-increasing table size.

Compared to **Replace Entire table**, **Replace Simulating truncate** offers an advantage in that it can maintain constraints such as check constraints that cannot be defined in metadata for SAS Data Integration Studio. If a target table is to have check constraints, the physical table must be created with all constraints applied before a Table Loader can load it with **Replace Simulating truncate**. This can be done once, outside of SAS Data Integration Studio or in user-written code in a SAS Data Integration Studio job. When the Loader step runs and the target table already exists, the step simulates a Truncate by creating an empty table with structure and constraints that are identical to the original, and then appends or inserts the data from the source table.

It is important to understand that **Replace Simulating truncate**, by design, ignores all constraint metadata when code is generated (except to create code to initialize the target if it does not already exist). Therefore, constraints on the physical table cannot be modified by changing constraint metadata and regenerated and rerunning with **Replace Simulating truncate**.

*Note:* If you are using Generation Data Sets, use the **Simulating Truncate** load technique instead of the DELETE \* syntax.

### 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. (It is 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 do not incur the overhead that is 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 can take unmatched records and add them to the target table during the same pass-through on 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 vary by hardware and operating environment, 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 requires reloading.

You might also want to temporarily remove key constraints in order to improve performance. If you remove constraints from the target before the load, then you remove the overhead of maintaining those constraints. If you are loading a significant number of transactions with data that conforms to the constraints, then removing the constraints should improve your performance.

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 four different tasks: put on, take off, leave as is, or recreate 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 added or not removed as needed. Select **On** for the **After Load** options to have indexes added after the load.

In some situations, you might select **Leave Off** in the **After Load** field to leave the indexes off during and after the table loading for performance reasons. One scenario is when the table is updated multiple times in a series of load steps. 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 large data volumes into a relational database.

### **Solution**

You should consider using the optimized SAS/ACCESS engine bulk loaders to bulk load the data into database tables. Many of the SAS/ACCESS engines for DBMS support the BULKLOAD option, and this loading capability is one of the fastest ways to insert large data volumes into a relational 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 periodically issuing a COMMIT. If you specify BULKLOAD=YES as a data set or a LIBNAME option, a database bulk-load method is used. This can significantly enhance performance, especially when database tables are indexed.

Consult SAS documentation to determine whether the BULKLOAD option is supported for your target database type and whether it can be specified as a LIBNAME or a data set option. For each database there are additional options to specify behavior of the bulkload option. These options can be found in the SAS/ACCESS documentation for the specific database. The names of these options normally start with BL\_.

Perform one of the following tasks to specify the BULKLOAD option:

- [“Set the BULKLOAD Option for a DBMS Library” on page 277](#)
- [“Set the BULKLOAD Option for a DBMS Table” on page 278](#)

### **Tasks**

#### **Set the BULKLOAD Option for a DBMS Library**

Some SAS/ACCESS engines allow you to specify the BULKLOAD option on the library. The LIBNAME statement enables you to assign a libref to a relational DBMS. This feature lets you reference a DBMS object directly in a DATA step or SAS procedure. You can use

it to read from and write to a DBMS object as if it were a SAS data set. You can associate a SAS libref with a relational DBMS database, schema, server, or group of tables and views.

The following DBMSs support BULKLOAD on the library level:

- ODBC
- OLE DB
- Teradata

Perform the following tasks to set the BULKLOAD= LIBNAME option:

1. Open the **Properties** window on the library icon, and select the **Options** tab.
2. Click on the **Advanced Options** button and select the **Output** tab.
3. Select **Yes** for the field labeled **Whether to use DBMS's bulk load**.

### ***Set the BULKLOAD Option for a DBMS Table***

You can specify the BULKLOAD option to load on an individual table level by using the data set option. This data set option applies only to the data set on which it is specified, and it remains in effect for the duration of the DATA step or procedure.

The DBMSs that support BULKLOAD on the table level are:

- DB2 UNIX for PC
- DB2 for z/OS
- Neoview
- Netezza
- ODBC
- OLE DB
- Oracle
- Sybase
- Teradata

Perform the following tasks to set the BULKLOAD= data set option:

1. Open the **Properties** window on the table icon and select the **Physical Storage** tab.
2. Click on the **Table Options** button.
3. Enter **BULKLOAD=YES** in the field labeled **Table options**.

## Chapter 16

# Working with SAS Sort Transformations

---

<b>About Sort Transformations</b> . . . . .	<b>279</b>
<b>Optimizing Sort Performance</b> . . . . .	<b>279</b>
Problem . . . . .	279
Solution . . . . .	280
<b>Creating a Table That Contains the Sorted Contents of a Source</b> . . . . .	<b>282</b>
Problem . . . . .	282
Solution . . . . .	282
Tasks . . . . .	282

---

## About Sort Transformations

The Sort transformation provides a graphic interface for the functions that are 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.

The properties window for the Sort transformation contains tabs that enable you to select the columns that you sort by and to set options for the sort. You can also optimize sort performance, as described in [“Optimizing Sort Performance” on page 279](#). For an example of how you can use a Sort transformation, see [“Creating a Table That Contains the Sorted Contents of a Source” on page 282](#).

---

## 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/resources/papers/tnote/tnote\\_performance.html](http://support.sas.com/resources/papers/tnote/tnote_performance.html).

**Table 16.1** 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: <ul style="list-style-type: none"> <li>• Minimize row width.</li> <li>• Drop unnecessary columns.</li> <li>• Minimize pad bytes.</li> </ul>
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.

Technique	Notes
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.
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 that is 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: <ul style="list-style-type: none"> <li>• input data</li> <li>• SORT procedure utility file</li> <li>• output data</li> </ul>
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 are dependent on the input file size.
Size SORT procedure utility files	Consider a number of factors to size the SORT procedure utility files: <ul style="list-style-type: none"> <li>• sizing information of the input data</li> <li>• any pad bytes added to character columns</li> <li>• any pad bytes added to short numeric columns</li> <li>• pad bytes that align each row by 8 bytes (for SAS data sets)</li> <li>• 8 bytes per row overhead for EQUALS processing</li> <li>• per-page unused space in the SORT procedure utility files</li> <li>• multi-pass merge: doubling of SORT procedure utility files (or sort failure)</li> </ul>
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 Contains the Sorted 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 Sort transformation to sort the data in a source table and write it to a target table. The sample job includes the following tasks:

- “Create and Populate the Job” on page 282
- “Specify How to Sort Information in the Target” on page 283
- “Run the Job and View the Output” on page 283

### Tasks

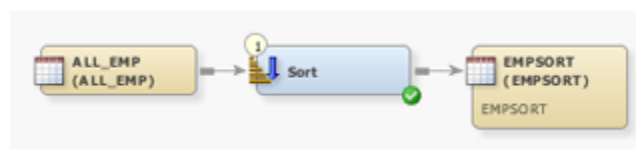
#### Create and Populate the Job

Perform the following steps to create and populate a new job:

1. Create an empty SAS Data Integration Studio job.
2. From the **Data** folder in the Transformations tree, select and drag a Sort transformation and drop it in the empty job on the **Diagram** tab in the Job Editor window.
3. Select and drag the source table from its folder and drop it before the Sort transformation on the **Diagram** tab.
4. Drag the cursor from the source table to the input port of the Sort transformation. This action connects the transformation to the source.
5. Because you want to have a permanent target table to contain the output for the transformation, right-click the temporary work table that is attached to the transformation and click **Replace** in the pop-up menu. Then, use the Table Selector window to select the target table for the job. The target table must be registered in SAS Data Integration Studio. (For more information about temporary work tables, see “Working with Default Temporary Output Tables” on page 126.)

The following example shows the sample process flow. The source table is named ALL\_EMP and the permanent target table is named EMPSORT.

**Display 16.1** Sample Sort Process Flow Diagram



### Specify How to Sort Information in the Target

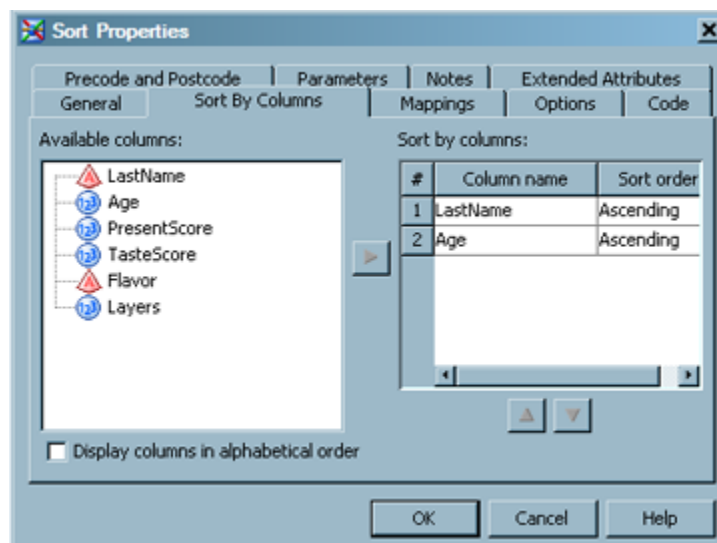
Perform the following steps to specify how to sort information in the target table:

1. Open the **Sort By Columns** tab of the properties window for the Sort transformation.
2. Select the first variable for the new sort from the list in the **Available Columns** field. 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.

*Note:* You can double-click on the value in the Sort order column to change the value. However, if you double-click on the value in the Column name column, the column is removed from the Sort by columns list.

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 Sort Tab for Sample Job

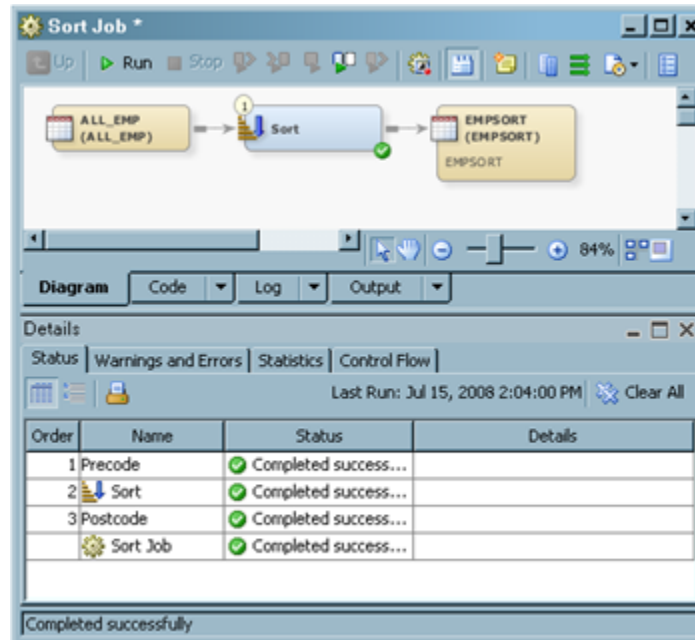


*Note:* Additional sorting options can be specified on the **Options** tab.

### Run the Job and View the Output

Perform the following steps to run the job and view the output:

1. Save the selection criteria for the target and close the properties window.
2. Right-click on an empty area of the job, and click **Run** in the pop-up menu. SAS Data Integration Studio generates code for the job and submits it to the SAS Application Server for execution. The following display shows a successful run of a sample job.

**Display 16.3** Successfully Completed Sample Job

3. If error messages are displayed on the **Status** tab, read and respond to the messages as needed.
4. To view the target table, right-click the target and select **Open**. The following display shows the target table data for the sample job.

**Display 16.4** Data in Sample Sorted Table

#	Age	Height	Weight	Name	Sex
1	11	51.3	50.5	Joyce	F
2	12	59.8	84.5	Jane	F
3	12	56.3	77	Louise	F
4	13	56.5	84	Alice	F
5	13	65.3	98	Barbara	F
6	14	62.8	102.5	Carol	F
7	14	64.3	90	Judy	F
8	15	62.5	112.5	Janet	F
9	15	66.5	112	Mary	F
10	11	57.5	85	Thomas	M
11	12	57.3	83	James	M
12	12	59	99.5	John	M

You can review the View Data window to ensure that the data from the source table was properly sorted. Note that the Age and Sex columns in the sample target table are sorted, but the other columns remained unsorted.



## Chapter 17

# Working with SQL Join Transformations

---

<b>About SQL Join Transformations</b> . . . . .	<b>287</b>
<b>Using the Designer Window</b> . . . . .	<b>287</b>
Problem . . . . .	287
Solution . . . . .	287
Tasks . . . . .	288
<b>Reviewing and Modifying Clauses, Joins, and Tables in an SQL Query</b> . . . . .	<b>288</b>
Problem . . . . .	288
Solution . . . . .	289
Tasks . . . . .	289
<b>Understanding Automatic Joins</b> . . . . .	<b>291</b>
The Autojoin Process . . . . .	291
A Sample Auto-Join Process . . . . .	292
<b>Selecting the Join Type</b> . . . . .	<b>294</b>
Problem . . . . .	294
Solution . . . . .	294
Tasks . . . . .	294
<b>Adding User-Written SQL Code</b> . . . . .	<b>296</b>
Problem . . . . .	296
Solution . . . . .	296
Additional Information . . . . .	297
<b>Debugging an SQL Query</b> . . . . .	<b>297</b>
Problem . . . . .	297
Solution . . . . .	298
Tasks . . . . .	298
<b>Adding a Column to the Target Table</b> . . . . .	<b>299</b>
Problem . . . . .	299
Solution . . . . .	299
Tasks . . . . .	299
<b>Adding a Join to an SQL Query on the Designer Tab</b> . . . . .	<b>299</b>
Problem . . . . .	299
Solution . . . . .	299
Tasks . . . . .	300
<b>Creating a Simple SQL Query</b> . . . . .	<b>301</b>
Problem . . . . .	301
Solution . . . . .	301
Tasks . . . . .	301

<b>Configuring a SELECT Clause</b> .....	<b>303</b>
Problem .....	303
Solution .....	303
Tasks .....	304
<b>Adding a CASE Expression</b> .....	<b>305</b>
Problem .....	305
Solution .....	306
Tasks .....	306
<b>Creating or Configuring a WHERE Clause</b> .....	<b>307</b>
Problem .....	307
Solution .....	307
Tasks .....	308
<b>Adding a GROUP BY Clause and a HAVING Clause</b> .....	<b>309</b>
Problem .....	309
Solution .....	309
Tasks .....	310
<b>Adding an ORDER BY Clause</b> .....	<b>312</b>
Problem .....	312
Solution .....	312
Tasks .....	312
<b>Adding Subqueries</b> .....	<b>313</b>
Problem .....	313
Solution .....	313
Tasks .....	314
<b>Validating or Submitting an SQL Query</b> .....	<b>318</b>
Problem .....	318
Solution .....	318
Tasks .....	318
<b>Joining a Table to Itself</b> .....	<b>319</b>
Problem .....	319
Solution .....	319
Tasks .....	319
<b>Using Parameters with an SQL Join</b> .....	<b>320</b>
Problem .....	320
Solution .....	320
<b>Constructing a SAS Scalable Performance Data Server Star Join</b> .....	<b>321</b>
Problem .....	321
Solution .....	321
Tasks .....	321
<b>Optimizing SQL Processing Performance</b> .....	<b>322</b>
Problem .....	322
Solution .....	322
<b>Performing General Data Optimization</b> .....	<b>323</b>
Problem .....	323
Solution .....	323
Tasks .....	323
<b>Influencing the Join Algorithm</b> .....	<b>324</b>
Problem .....	324
Solution .....	324
Tasks .....	324

<b>Setting the Implicit Property for a Join</b> .....	<b>325</b>
Problem .....	325
Solution .....	325
<b>Enabling Pass-Through Processing</b> .....	<b>326</b>
Problem .....	326
Solution .....	327
Tasks .....	327
<b>Using Properties Window Options to Optimize SQL Processing</b>	
<b>Performance</b> .....	<b>328</b>
Problem .....	328
Solution .....	329
Tasks .....	329

---

## 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 supports joining the table to itself. It also supports subqueries; the CASE expression; and WHERE, GROUP BY, HAVING, and ORDER BY clauses.

The process of building the SQL query is performed in the Designer window. You access this window when you double-click the SQL Join transformation in a SAS Data Integration Studio job. You use the Designer window to create, edit, and review an SQL query. The window contains sections that are designed to simplify creating the SQL query and configuring its parts. To return to the SQL job on the **Designer** tab of the Job Editor window, click **Up** on the toolbar.

---

## Using the Designer Window

### **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 window for the SQL transformation to create, edit, and review an SQL query. You access this window when you double-click the SQL Join in a SAS Data Integration Studio job. (You can also right-click the transformation and click **Open** in the pop-up menu.) The window contains sections that are designed to simplify creating the SQL query and configuring its parts.

## Tasks

### Using Components in the Designer Window

The Designer window enables you to perform the tasks listed in the following table:

**Table 17.1** Designer Tab Tasks

Task	Location	Action
Select and manipulate an object that displays in the <b>Diagram</b> tab.	Navigate pane	Click the object that you need to access.
Add SQL clauses to the flow shown on the <b>Diagram</b> tab.	SQL Clauses pane	Double-click the clause or drop it on the <b>Diagram</b> 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 <b>Display columns in alphabetical order</b> .	Tables pane	Click <b>Select</b> , <b>Where</b> , <b>Having</b> , <b>Group by</b> , or <b>Order by</b> in the SQL Clauses pane.
Display and update the main properties of an object that is selected on the <b>Diagram</b> tab. The title of this pane changes to match the object selected in the Navigate pane.	Properties pane	Click an object on the <b>Diagram</b> tab.
Create SQL statements, configure the clauses that are 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.	<b>Diagram</b> tab	Click <b>SQL Join</b> , <b>Create</b> , or <b>From</b> in the Navigate pane.
View the SAS code generated for the query.	<b>Code</b> tab	Click <b>Code</b> at the bottom of the <b>Diagram</b> tab.
View the log of a SAS program, such as the code that is executed or validated for the SQL query.	<b>Log</b> tab	Click <b>Log</b> at the bottom of the <b>Diagram</b> tab.

---

## 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 window for the SQL transformation to access and review the objects in your query.

Perform the following tasks:

- [“Review Clauses, Join, and Tables” on page 289](#)
- [“Modify Properties of Clauses and Tables” on page 290](#)

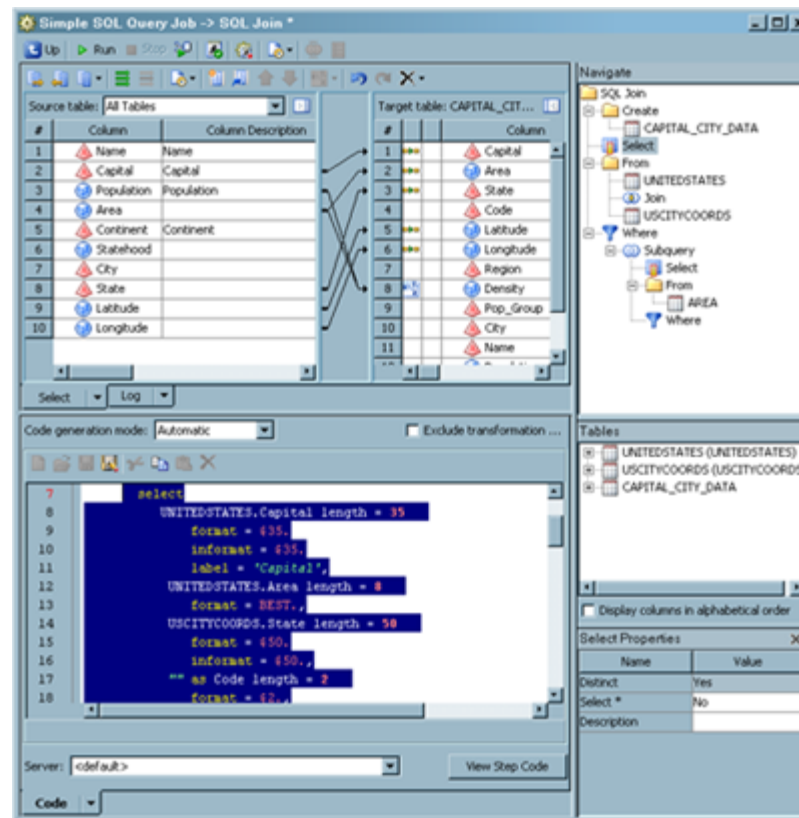
## **Tasks**

### ***Review Clauses, Join, and Tables***

When you click an item in the Navigate pane, the Designer window 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 in a tab on the left side of the Designer window. When you click a table, the columns from the table are shown in a tab.
- 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 window for a sample job.

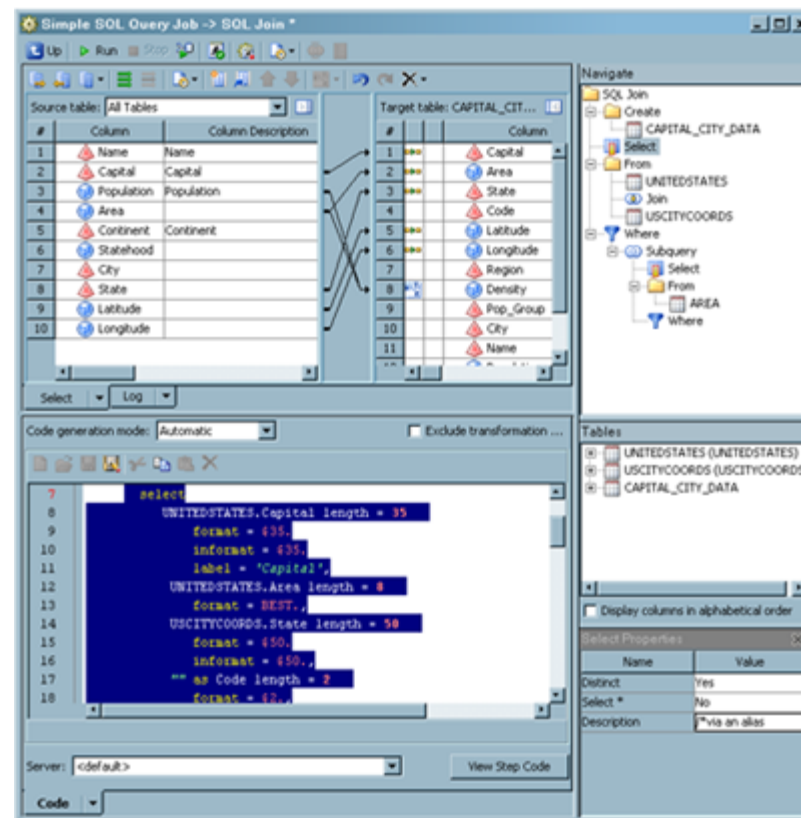
**Display 17.1** Information about a Select Clause on a Designer Tab

Note that **Select** is highlighted on the Navigate pane, and the SQL code for the SELECT clause is highlighted on the **Code** tab. To highlight the code for a query object, right-click the object in the Navigate pane and click **Find In**. Then, click **Code** in the submenu. 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 to modify the object directly. If the properties window is not displayed, click **Show Properties Pane** in the toolbar at the top of the Designer window.

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 **Code** tab. See the following display for a sample view of this behavior.

**Display 17.2** Using the Description Field to Comment a Select Clause

Note that text entered in the **Description** field in the Select Properties pane is also displayed immediately before the SQL code on the **Code** tab. If you were to delete the text from the **Description** field, it would also be removed from the Navigate pane and the **Code** tab. Once again, you highlight the code with the **Find In** pop-up menu option. 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 Autojoin Process

The automatic join (auto-join) process determines the initial relationships and conditions for a query that is formulated in the SQL Join transformation. You can understand how these relationships and conditions are established. You can also examine how port 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 that are added to SQL transformation as input. When more than one table is connected to 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 connected and making it the left side of the join. Then, the next table connected becomes the right side. If more than two tables are connected, 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 locates all of the tables that are connected to the join. This process continues until it includes all of the joins that are connected to the first join.

The auto-join process is geared toward finding the best relationships between the tables. This process is based on the known relationships that are documented as key constraints, indexes, or both. 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 by using indexes alone, but an index-only match can occur only when columns 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.

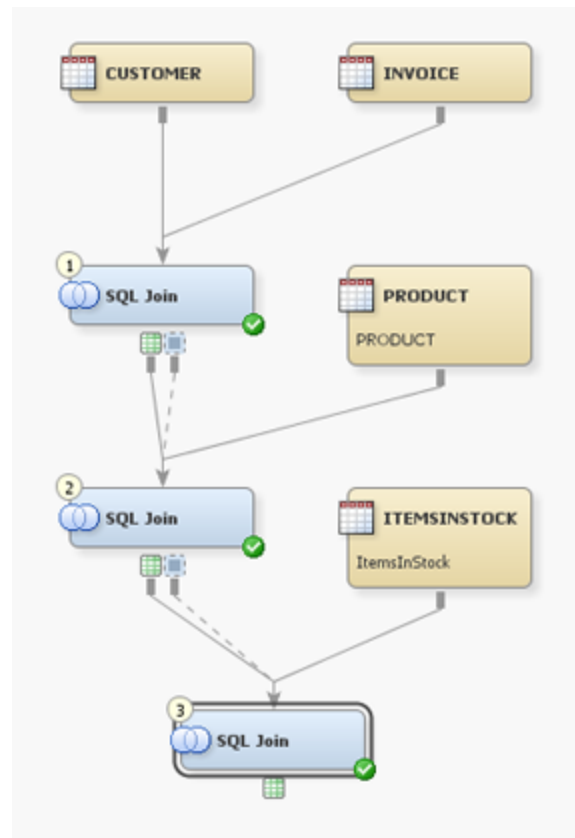
### **A Sample Auto-Join Process**

An auto-join is best explained with a specific example. Suppose you add the following tables as input to the SQL Join transformation in the following order:

- CUSTOMER, 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: ITEMSINSTOCK
- PRODUCT, with the following constraint defined:
  - Primary key: ITEMSINSTOCK
- ITEMSINSTOCK, with the following constraint defined:
  - Index: ITEMSINSTOCK

After the auto-join process is run for this source data, the process flow that is depicted in the following display is shown in the **Diagram** tab in the Designer 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: CUSTOMER, INVOICE, PRODUCT, and ITEMSINSTOCK. This flow means that the join at the top of diagram is created first, followed by the join in middle. Finally, the join at the bottom is created. 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 all of the tables 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 CUSTOMER 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 CUSTOMER and INVOICE tables as the new left table and PRODUCT as the right table. A key relationship between INVOICE and PRODUCT on the column ITEMSINSTOCK 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 PRODUCT table as the new left table and ITEMSINSTOCK as the right table. No key relationships are found, so the indexes are searched. A match is found between PRODUCT and INVENTORY on the column ITEMSINSTOCK. Both tables are then removed from the search and the matching process is finished.

The relationship is initialized as follows:

```
CUSTOMER.CUSTOMER_ID = INVOICE.CUSTOMER_ID and
INVOICE.ITEMSINSTOCK = PRODUCT.ITEMSINSTOCK and
PRODUCT.ITEMSINSTOCK = ITEMSINSTOCK.ITEMSINSTOCK
```

---

## 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 that is 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 to select a different join type.

### Tasks

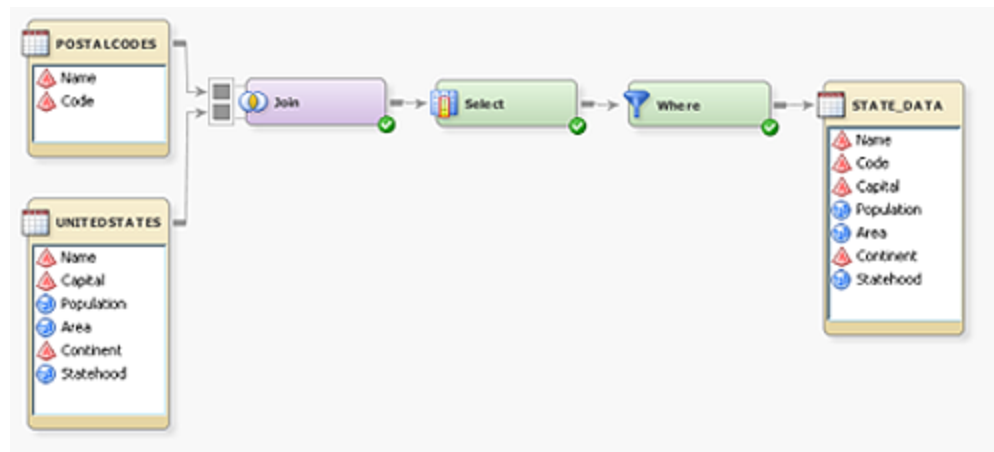
#### ***Change Join Types in a Sample SQL Query***

Examine a sample SQL query in a SAS Data Integration Studio job to see the effects of changing the join types that are used in the query. The sample query contains the tables and columns that are listed in the following table:

**Table 17.2** Sample Query Data

Source Table 1: POSTALCODES	Source Table 2: UNITEDSTATES	Target Table: State_Data
Name	Name	Name
Code	Capital	Code
	Population	Capital
	Area	Population
	Continent	Area
	Statehood	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 or 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
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

Join Type	Description	Data Included in Results	Implicit or Explicit Status
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
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	...	...	...	...	...

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

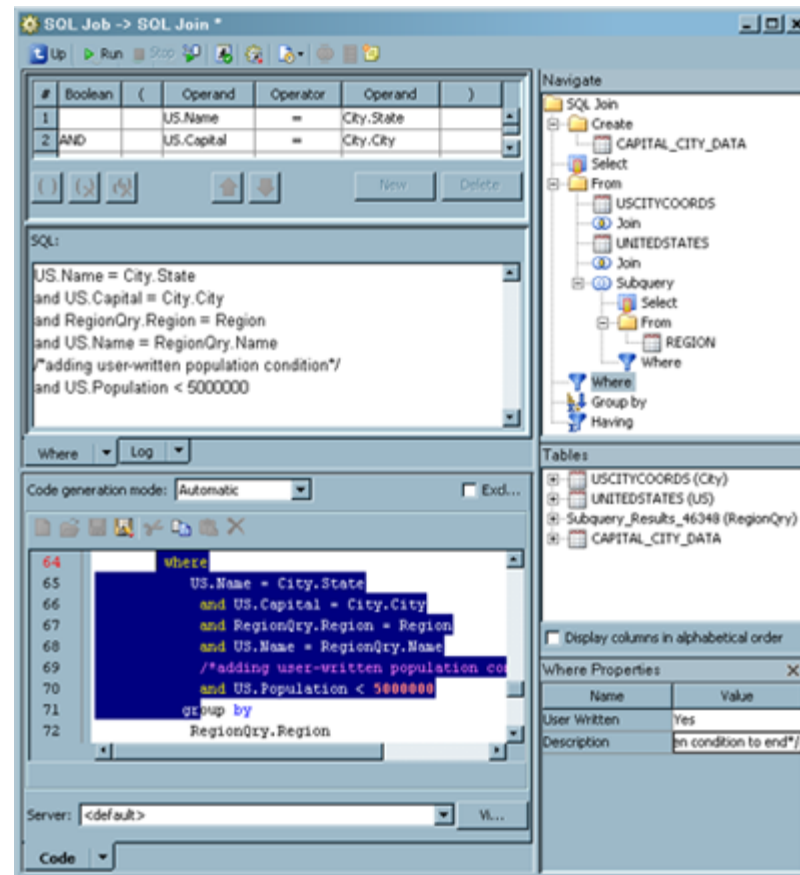
You want to add user-written code to an SQL query that is used in a SAS Data Integration Studio job. This user-written code can consist of SQL code that is added to a WHERE, HAVING, or JOIN clause. It can also overwrite the entire DATA step for the SQL Join transformation.

### 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 in the Designer window. 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



Note that the following line of SQL code was added to the **SQL** field on the **Where** tab:

```
and US.Population < 5000000
```

This code is also highlighted on the **Code** tab.

### Additional Information

For information about how to overwrite the entire **DATA** step for the SQL Join transformation, see [“About User-Written Code” on page 213](#).

## 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. Perform the following tasks:

- “Set the Debug Property” on page 298
- “Examine Some Sample Method Traces” on page 298

## 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 perform this task, 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 column** 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 on 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 **Diagram** tab in the Designer window. You can easily tie this new join into the existing query flow.

## Tasks

### Add a Join to the Diagram Tab

Perform the following steps to add a join to the **Diagram** tab:

1. Select one of the join objects in the **Joins** folder in the SQL Clauses pane, and drop it in a blank space on the **Diagram** tab.
2. Disconnect the existing join from the Select object. Click on the arrow between the Join and the Select object. Then, press DELETE to remove the arrow. The new join and the original join are displayed in the query flow, as shown in the following display.

**Display 17.7** Initial Job Flow



3. Move the new join to an appropriate location. Then, complete the following actions:

- Connect the original join to one input port of the new join.

*Note:* If you select a Join node on the diagram, then the new join node will be inserted after the join that you selected.

- Drop the source table for the new join onto the **Diagram** tab.
- Connect the table to the remaining input port of the new join.
- Connect the new join to the input port of the Select object.

*Note:* If you select the Select node on the diagram, then the join is automatically connected or inserted between the Select node and the Join node.

A sample job that includes an added join is shown in the following display.

**Display 17.8** Added Join



*Note:* You can add the source and target tables directly to the process flow diagram for the job in the **Diagram** tab for the Job Editor window. You can also add a table, join, or subquery to a job by dragging and dropping it on the **Diagram** tab in the Designer window for the SQL Join transformation.



---

## 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 to create an SQL query that runs in the context of a SAS job. 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.

Perform the following tasks:

- [“Create and Populate the Job” on page 301](#)
- [“Create the SQL Query” on page 302](#)

### Tasks

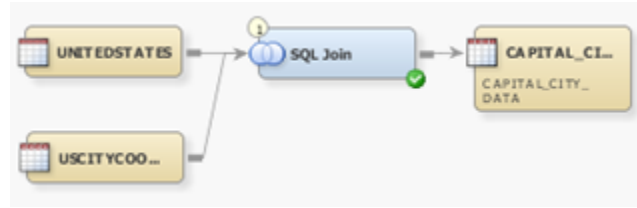
#### Create and Populate the Job

Perform the following steps to create and populate the job:

1. Select and drag an SQL Join transformation from the **Data** folder in the Transformations tree. Then, drop it in the empty job on the **Diagram** tab in the Job Editor window.
2. Select and drag the source tables out of the Inventory tree. Then, drop it before the SQL Join transformation on the **Diagram** tab. Drag the cursor from the source tables to the input port of the SQL Join transformation. This action connects the sources to the transformation.
3. Select and drag the target table out of the Inventory tree. Then, drop it after the SQL Join transformation on the **Diagram** tab.
4. Because you want to have a permanent target table to contain the output for the transformation, right-click the temporary work table that is attached to the SQL Join transformation and click **Replace** in the pop-up menu. Then, use the Table Selector window to select the target table for the job. The target table must be registered in SAS Data Integration Studio. (For more information about temporary work tables, see [“Working with Default Temporary Output Tables” on page 126.](#))

*Note:* If you keep the worktable, you must add the Table Loader transformation to the job in order to connect the target table into the job flow. The Table Loader provides additional load options and combinations of load options, but it is not needed for many jobs. The extra processing that is required for the Table Loader can degrade performance when the job is run. In addition, you should not use a temporary output table and a Table Loader step if you use pass-through processing when your target table is a DBMS table and your DBMS engine supports the **Create as Select** syntax.

The following display shows a sample SQL job.

**Display 17.9** Sample SQL Job

*Note:* The source tables for the sample job are UNITEDSTATES and USCITYCOORDS. The target table is named CAPITAL\_CITY\_DATA. 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:

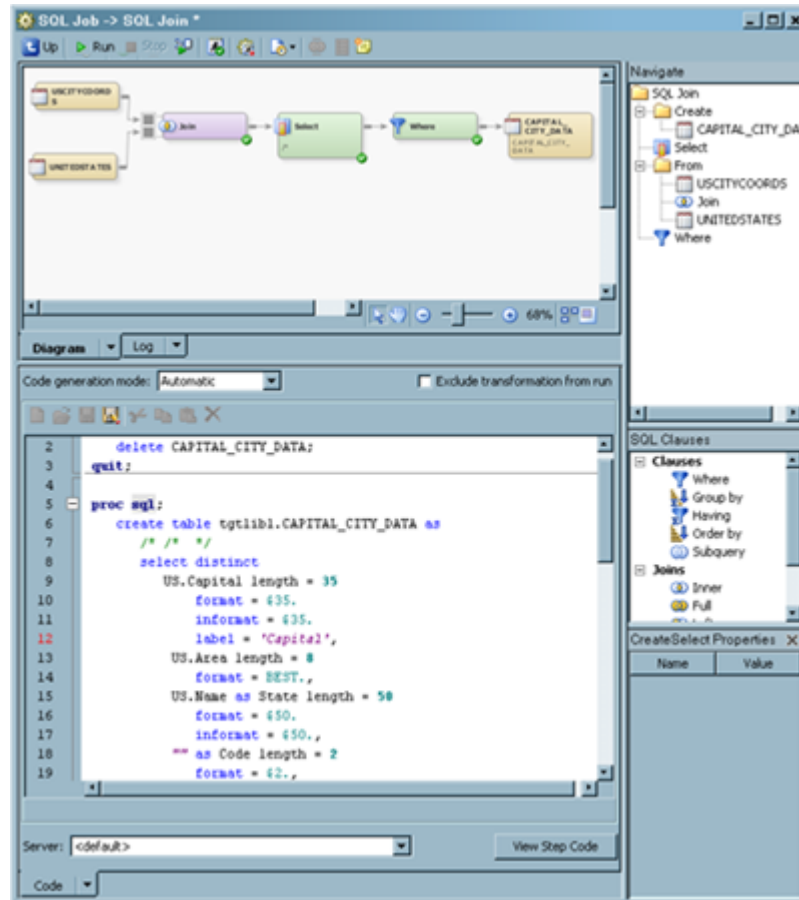
1. Double-click the SQL Join transformation to open the Designer window.
2. Click **SQL Join** in the Navigate pane. The right-hand side of the Designer window contains a Navigate pane, an SQL Clauses/Tables pane, and a properties pane. You might need to resize the horizontal borders of the panes to see all three of them. For more information, see [“Using the Designer Window” on page 287](#).

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 that are output from the job in the **Max Output Rows** field.

3. Click **Create** in the Navigate pane to display an initial view of the query on the **Diagram** tab. Note that the sample query already contains an INNER join, a SELECT statement, 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 actually joins the tables. However, they do reflect the SQL syntax.

You can click the tables that are included in the query and set an alias in the properties pane for each. These aliases help simplify the SQL code that is generated in the query. Aliases are set for the source tables in the sample job. The Designer window is shown in the following display.

Display 17.10 Sample Designer Tab



*Note:* 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. 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 **Code** tab.

## 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 are read from the source tables and which columns are 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 in the Designer window for the SQL Join transformation.

## Tasks

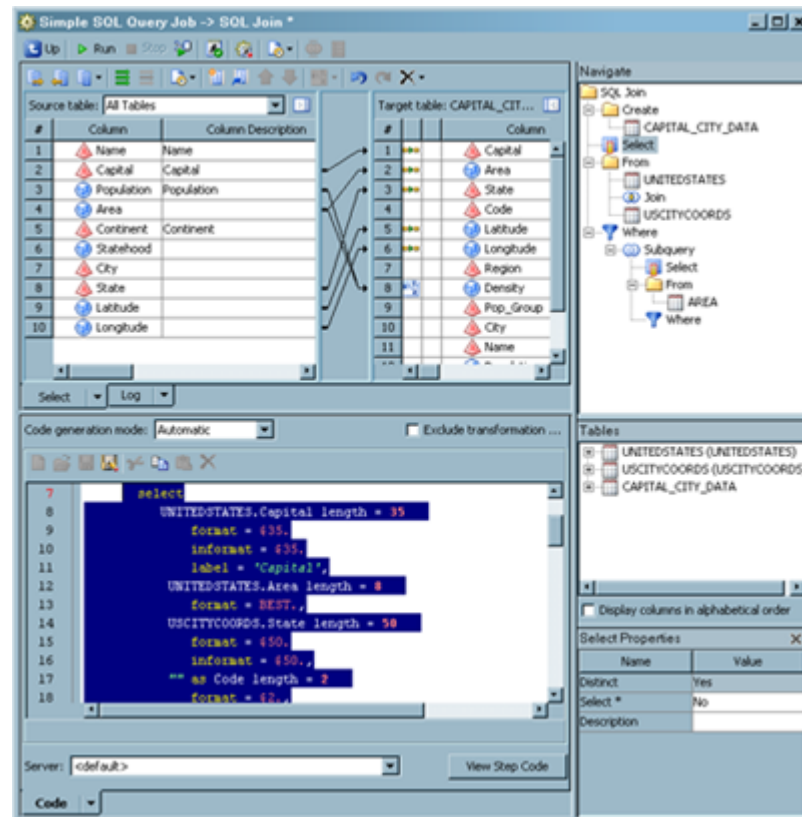
### Configure the **SELECT** Clause with the **Select Tab**

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 between the **Source table** and **Target table** fields. Then, click **Map All** 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** column in the **Target table** field, and click **Advanced**. The Expression Builder window displays. For information about the Expression Builder window, see [“Expression Builder” on page 402](#).
  - Enter the expression that you need to create into the **Expression Text** field. (You can use the **Data Sources** tab to navigate to the column names.) Click **OK** to close the window.
  - Review the data in the row that contains the derived expression. Ensure that the column formats are appropriate for the data that is generated by the expression. Change the formats as necessary.

To highlight the code for the Select object, right-click the object in the Navigate pane and click **Find In**. Then, click **Code** in the submenu. The following display depicts a sample **Select** tab.

Display 17.11 Sample Select Tab Settings



4. Review the data tables in the **Source table** field and the **Target table** field to avoid mapping errors. For example, the Name column in the US source table uses the full names of the states, such as California. However, the State column in the CITY target table uses the two-letter state abbreviation (CA). In this case, the column width for the State column must be increased to 50 in order to accommodate the data in the Name column. Also, the **Distinct** property in the Select Properties pane is set to *Yes*. This property determines that only the first matching record for each matching condition is included in the output. Note that the SQL code for the SELECT clause is highlighted on the **Code** 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.

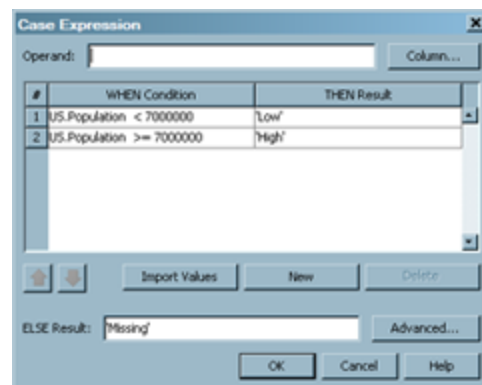
## Tasks

### Add a CASE Expression to an SQL Query in the Designer Window

Perform the following steps to add a CASE expression to the SQL query in the Designer window:

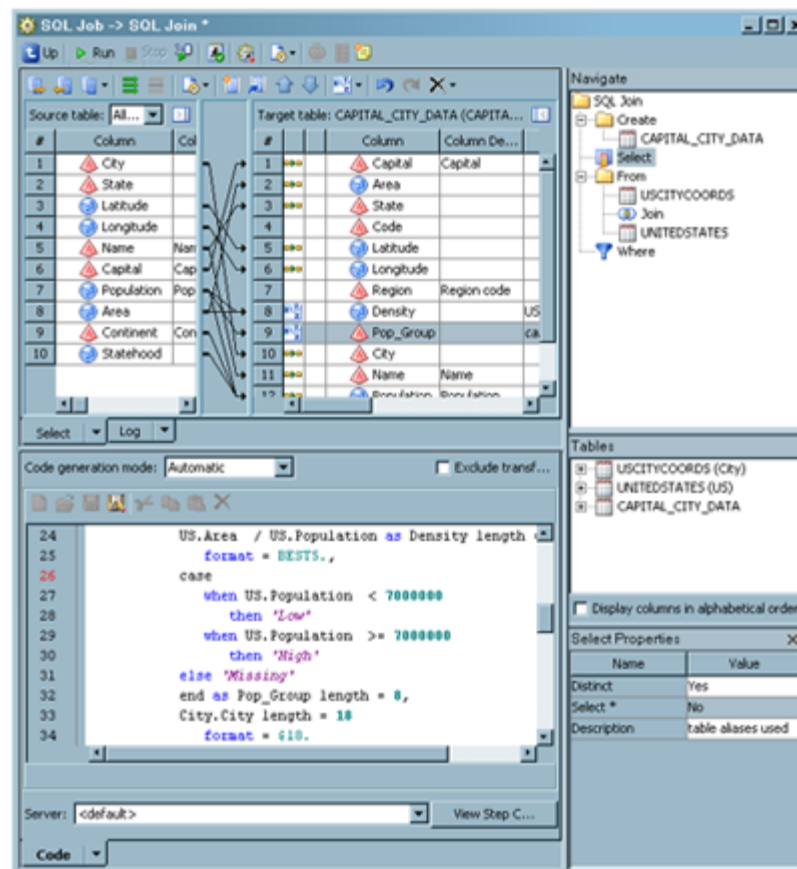
1. Access the CASE Expression window. To perform this task, click **CASE** in the drop-down menu for an Operand in a WHERE, HAVING, or JOIN condition. You can also access the **CASE** option in the Expression column for any column that is listed in the **Target table** field on the **Select** tab.
2. 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 and THEN clause.
4. Add the remaining WHEN and 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 and 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 17.12** Sample Completed CASE Expression Window



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 US.Population column name must be entered for each WHEN condition in the expression. In the sample query, the CASE expression is added to a Pop\_Group column that has been added to the target table. The following display depicts the **Select** tab.

Display 17.13 Sample CASE Expression Query



Note that the Population column in the **Source table** field on 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.

*Note:* Make sure that the option in the **Select\*** field of the Select Properties pane is set to **No**. The CASE expression is not included in the SQL SELECT statement when this option is enabled.

## 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 window for the SQL Join transformation to configure the WHERE clause for an SQL query.

## Tasks

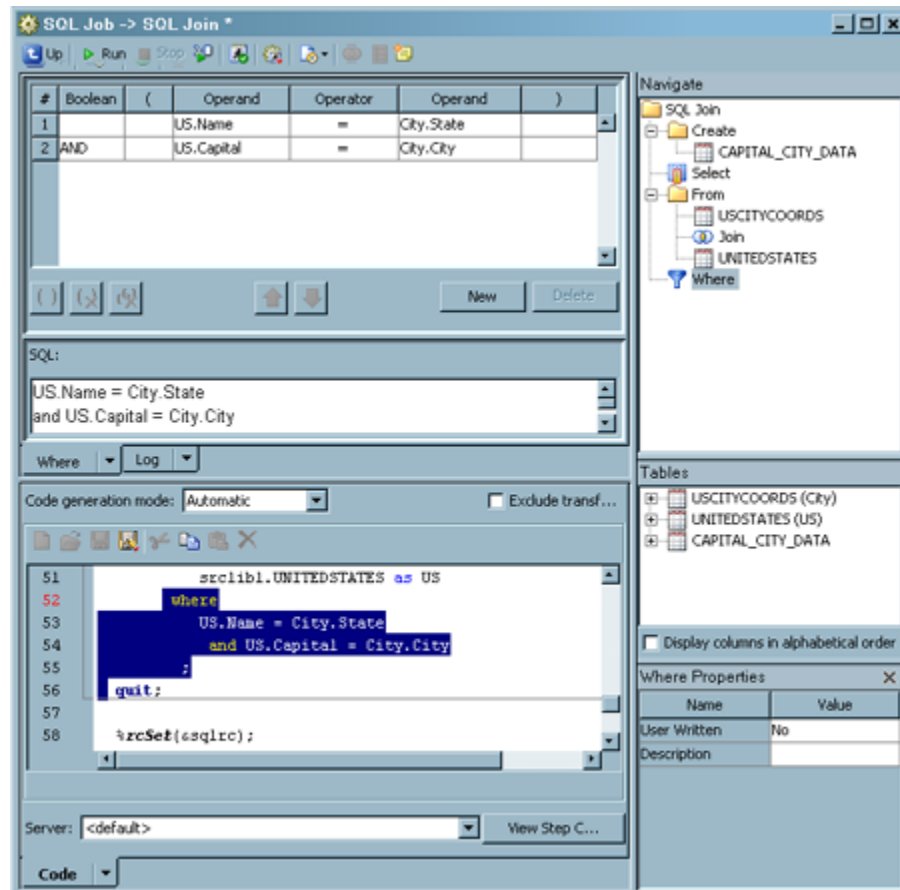
### Configure the WHERE Clause with the Where Tab

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 **Diagram** tab, double-click **Where** in the SQL Clauses pane. The **Where** clause object is added to the query flow in the **Diagram** tab. Note that **Where** clause objects are automatically populated into the **Diagram** 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.
5. Add the remaining conditions for the WHERE clause. You need to add one row for each condition.
6. The conditions created for the sample query are depicted in the SQL code that is generated in this step in the **SQL** field, as shown in the following display.



Display 17.14 Sample Where Tab Settings



Note that the SQL code for the WHERE clause that is shown in the **SQL** field is identical to the highlighted WHERE clause code that is displayed on the **Code** tab. To highlight the code for a query object such as the Where object, right-click the object in the Navigate pane and click **Find In**. Then, click **Code** in the submenu.

## 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.

Perform the following tasks:

- “Add a GROUP BY Clause to an SQL Query in the Diagram Tab” on page 310
- “Add a HAVING Clause to an SQL Query in the Diagram Tab” on page 310

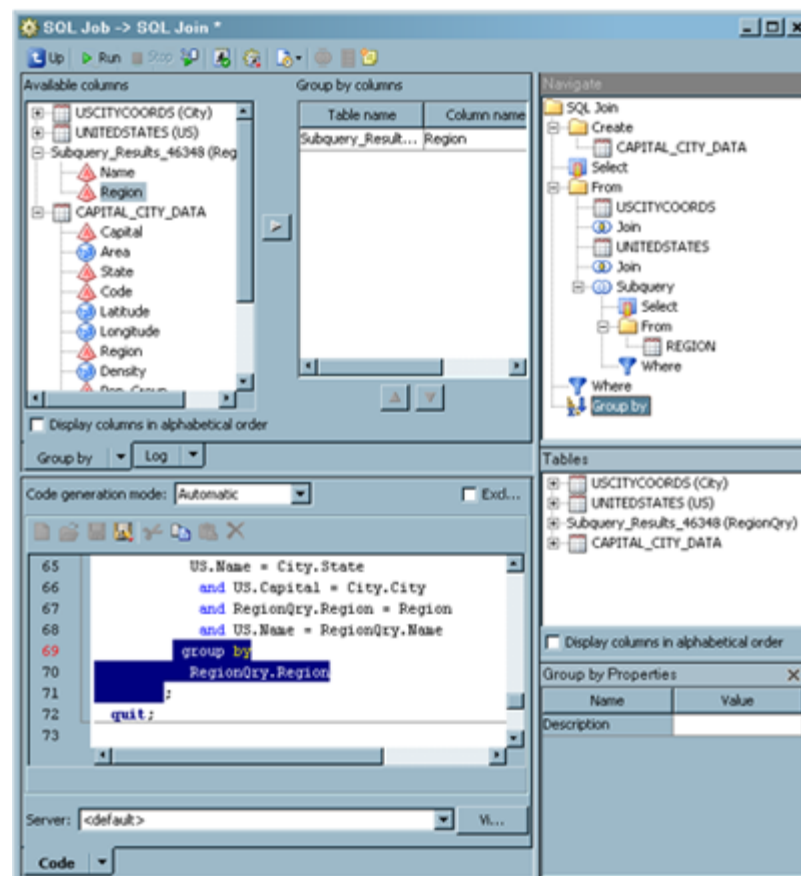
## Tasks

### Add a GROUP BY Clause to an SQL Query in the Diagram Tab

Perform the following steps to add a GROUP BY clause to the SQL query in the Diagram tab in the Designer window:

1. Click **Create** in the Navigate pane to access the **Diagram** 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 **Diagram** tab. Then, click **Group by** in the Navigate pane to access the **Group by** tab.
3. 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. The following display depicts a sample SQL query grouped with a GROUP BY clause.

**Display 17.15** Sample SQL Query Grouped with a GROUP BY Clause



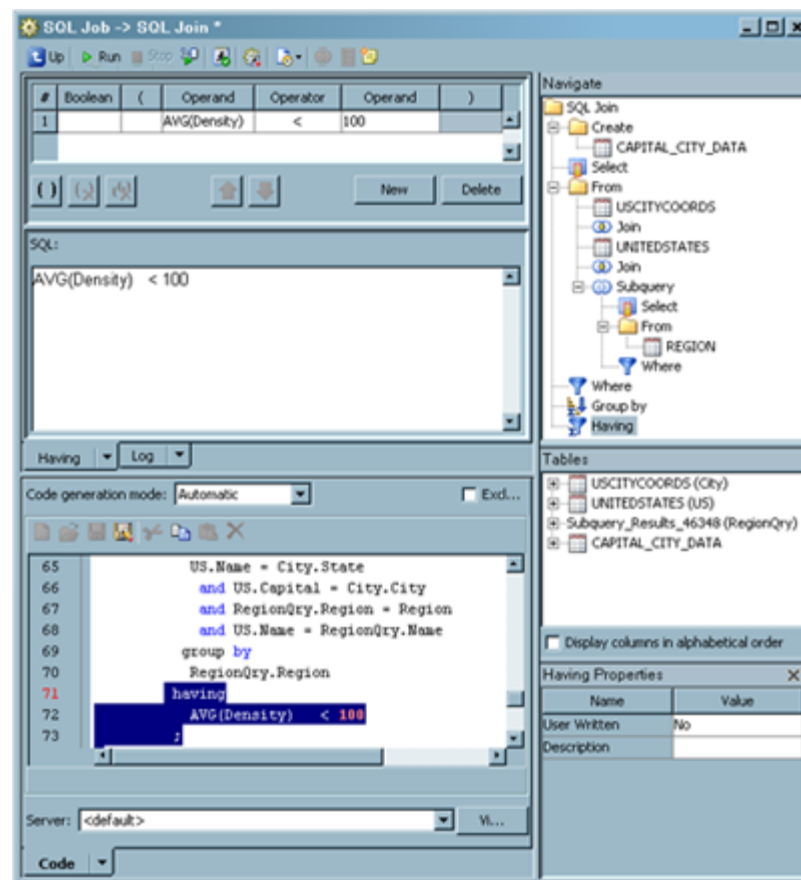
Note that the Group by column pane is set on the **Group by** tab, and the resulting SQL code is highlighted on the **Code** tab. The GROUP BY clause in the sample query groups the results of the query by the region of the United States.

### Add a HAVING Clause to an SQL Query in the Diagram Tab

Perform the following steps to add a HAVING clause to the SQL query in the **Diagram** tab in the Designer window:

1. Click **Create** in the Navigate pane to access the **Diagram** 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 **Diagram** 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.
6. Add the remaining conditions for the HAVING clause. You need to add one row for each condition.
7. The condition that is 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 17.16** Sample SQL Query Subsetted with a HAVING Clause



Note that the SQL code for the HAVING clause that is shown in the **SQL** field is identical to the highlighted HAVING clause code that is displayed on the **Code** tab. (To highlight the code for a query object, right-click the object in the Navigate pane and click **Find In**. Then, click **Code** in the submenu.) 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.

---

## 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**

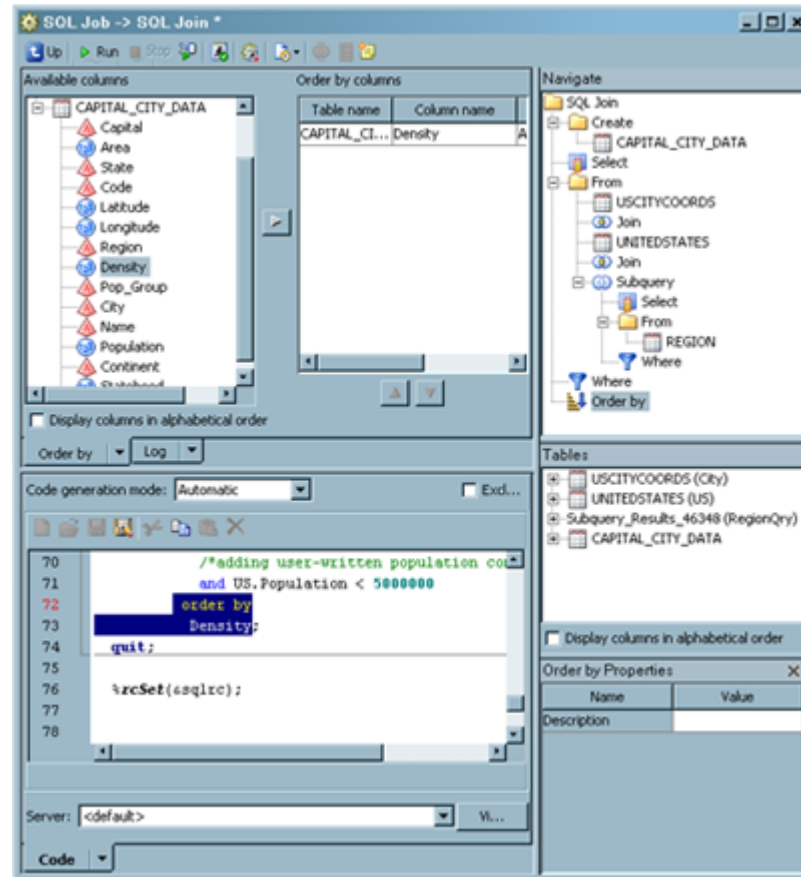
You can use the **Order by** tab in the Designer window to add an ORDER By clause to the SQL query.

### **Tasks**

#### ***Add an ORDER BY Clause to an SQL Query in the Diagram 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 window:

1. Click **Create** in the Navigate pane to access the **Diagram** 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 **Diagram** tab.
3. Click the **Order by** object in the SQL Clauses pane to access the **Order by** tab.
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. The following display depicts a sample SQL query with an ORDER BY clause.

**Display 17.17** Sample SQL Query Sorted with an ORDER BY Clause

Note that the ORDER BY column is set on the **Order by** tab, and the resulting SQL code is highlighted on the **Code** tab. To highlight the code for a query object, right-click the object in the Navigate pane and click **Find In**. Then, click **Code** in the submenu.

## 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 window to add a subquery to an SQL query. The sample job used in [“Add a Subquery As an Input Table” on page 314](#) 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 317](#) covers adding a subquery to a SELECT, WHERE, or HAVING clause in an SQL query.

Perform the following tasks:

- [“Add a Subquery As an Input Table” on page 314](#)

- “Add a Subquery to an SQL Clause” on page 317

## Tasks

### Add a Subquery As 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 the **Diagram** tab in the **Designer** window for the SQL Join transformation. If you drop a table on an existing table in the **Designer** tab, the new table replaces the existing table.

You can even add a new input port to the query flow on the **Diagram** 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 **Diagram** tab. The join and its input port is displayed in the query flow in the tab, where you can connect it to the appropriate parts of the SQL query. 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 the **SubQuery** object in the **Select Clauses** folder in the SQL Clauses pane, and drop it in a blank space in the **Diagram** tab.
2. Select the **Inner** join object in the **Joins** folder in the SQL Clauses pane, and drop it in a blank space in the **Diagram** tab.
3. Disconnect the existing join from the Select object. Click on the arrow between the Join and the Select object. Then, press DELETE to remove the arrow. The subquery, the inner join, and the original join are displayed in the query flow, as shown in the following display.

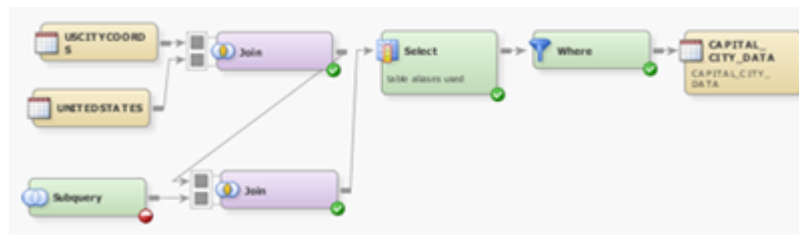
**Display 17.18** Initial Subquery on Inner Join



4. Move the subquery and the new join to appropriate locations. Then, complete the following actions:
  - Connect the subquery to an input port of the new join.
  - Connect the original join to the remaining input port of the new join.
  - Connect the new join to the input port of the Select object.

A sample subquery on an inner join is shown in the following display.

**Display 17.19** Connected Subquery On Inner Join

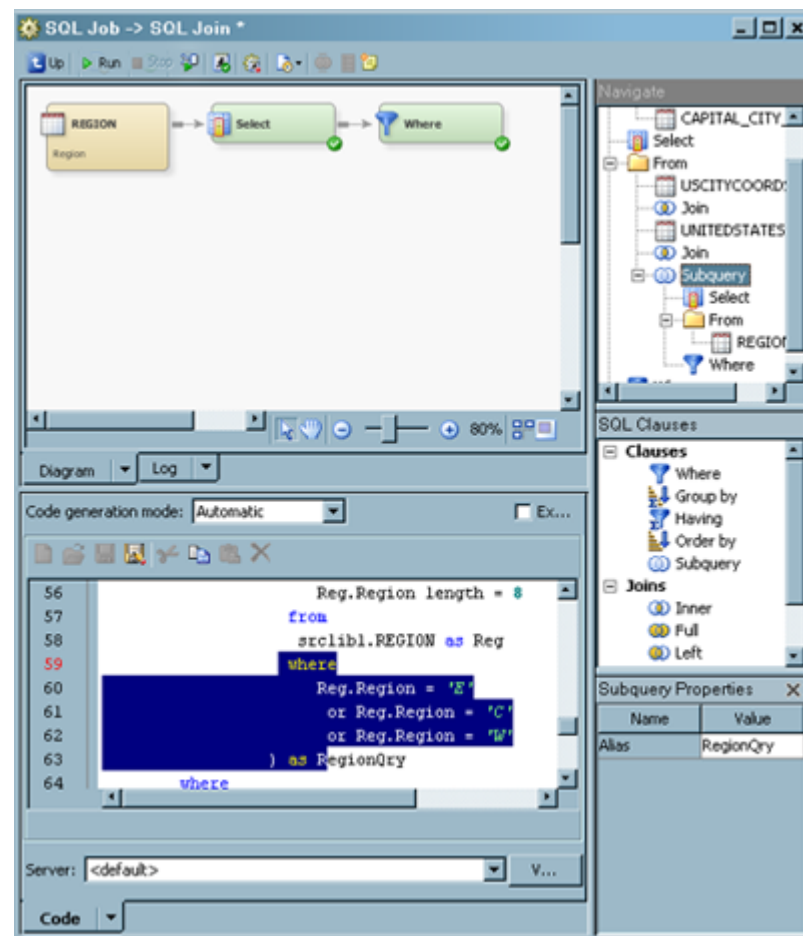


5. Click the **SubQuery** object. Note that the SubQuery Properties pane displays. Enter an appropriate value in the **Alias** field. (**RegionQry** was entered in the sample job.) If

you do not enter an alias here, then the subquery fails. The system-generated name for the subquery results table is too ambiguous to be recognized as an input to the full SQL query.

6. Click **SubQuery** in Navigate pane. The Select object for the Subquery is displayed on a **Diagram** tab.
7. Drop the source table on the **Diagram** tab. The source table for the sample job is named **Region**.
8. Double-click **Select** to display the **Select** tab. Make sure that the source table columns are mapped properly to the target table. Also, ensure that the **Select \*** property in the Select Properties pane is set to **No**.
9. 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 **Diagram** tab. A **Where** clause object is added to the **Diagram** tab. The completed subquery flow is shown in the following display.

**Display 17.20** Sample Subquery Flow

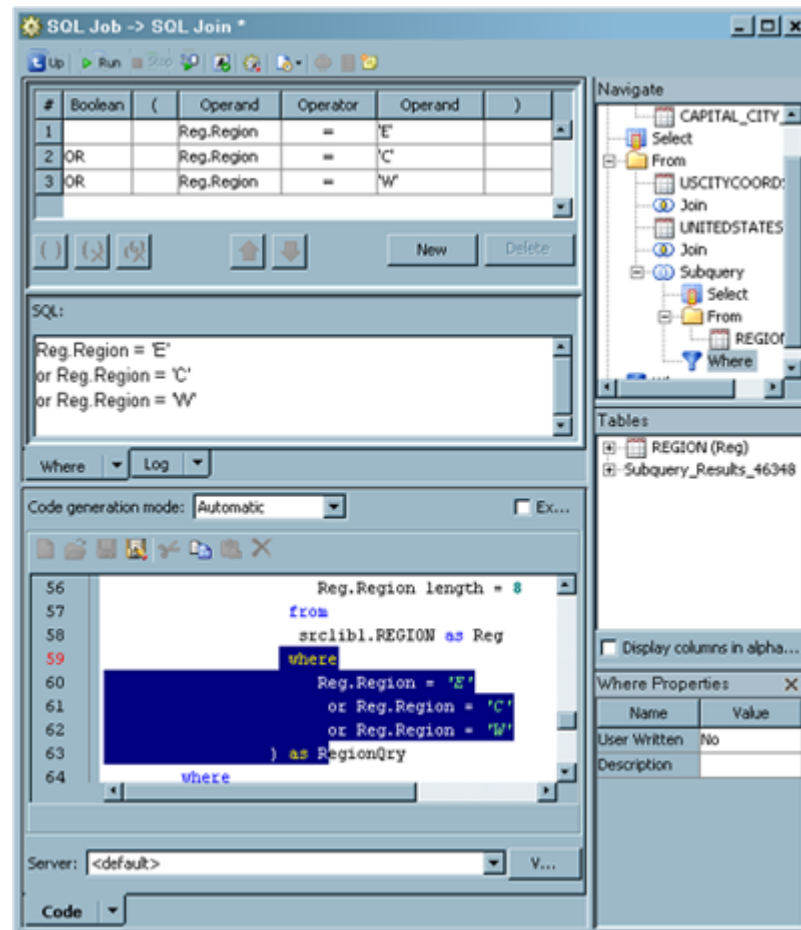


10. Double-click **Where** to display the **Where** tab.
11. Click **New** on the **Where** tab to begin the first part of the expression. An editable row appears in the table near the top of the tab.
12. Create your first WHERE condition. In this example, a subset of the **Region** column from the **Region** 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.

13. Keep the defaulted value of = in the **Operator** field. Enter the value `'E'` in the **Operand** field on the right side of the row.
14. Create the remaining conditions for the WHERE statement. Review the SQL code that is generated in this step in the SQL field, as shown in the following display.

**Display 17.21** Where Tab in the Subquery



15. A connection is required between the source table for the subquery and the target table for the full SQL 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.
16. Enter name of the subquery source table in the **Name** field. Then, make sure that the new column has the appropriate data type. In this case, the Region table is added to the target table in the SQL query.
17. Add a mapping for the subquery to the main query SELECT clause. In the sample query, the Region column from the Region table in the subquery is mapped to the Region column in the target table. Also, the following condition is added to the main query WHERE clause:

```
and RegionQry.Region = Region
```

This condition connects the inner join subquery to the main query.

*Note:* You can add a subquery to any place that you can add a table.



### 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 17.22** 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 17.23** Sample WHERE Clause Subquery

Simple SQL Query Job -> SQL Join \*

Diagram: AREA -> Select -> Where

Code generation mode: Automatic

```

44  src1tbl.USCITYCOORDS
45  where
46  EXISTS (
47  select *
48  from
49  src1tbl.AREA
50  where
51  UNITEDSTATES.Area = AREA.Area
52  and AREA.Area > 50000
53  )
54  ;
55  quit;

```

SQL Clauses:

- Where
- Group by
- Having
- Order by
- Subquery

Where Properties:

Name	Value
User Written	No
Description	

The subquery includes a source table, a SELECT clause, and a WHERE clause. You can compare the tree view of the subquery in the Navigate pane to the process flow on the **Diagram** tab and the code that is highlighted on the **Code** tab.

---

## Validating or Submitting an SQL Query

### Problem

You want to either validate that the code in an SQL query works properly when the SAS Data Integration Studio job that contains it is run at a later time or immediately submit the query as part of a job.

### Solution

You can validate the code in an SQL query in the Designer window for 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 as part of the SAS Data Integration Studio job that contains the SQL Join transformation.

- [“Validate the Code in an SQL Query” on page 318](#)
- [“Submit a Query As a Part of a SAS Data Integration Studio Job” on page 318](#)

### Tasks

#### **Validate the Code in an SQL Query**

Perform the following steps to validate a query in the Designer window:

1. Click **Validate SQL** in the toolbar at the top of the Designer window.
2. Examine the **Log** tab that is displayed in the Designer window to verify that the query was submitted successfully or to troubleshoot an unsuccessful submission.

*Note:* You can use the Runtime Manager in SAS Data Integration Studio to cancel the SQL query. The SQL Join transformation is displayed as a row in the Runtime Manager. You can right-click the row and click **Stop Job** to cancel the query. (You can also click **Stop** in the Designer window toolbar.) 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 **Run** on the SAS Data Integration Studio menu bar.
  - Right-click in the Job Editor window. Then, click **Run**.
  - Click **Run** on the SAS Data Integration Studio **Actions** menu.
2. Validate the job as needed. For example, you can check the properties of the target table. You can also review the data that is 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.

*Note:* You can click **Run to Selected Transform** on the Designer window toolbar to specify that only the steps that are placed before the SQL query code are submitted. (These steps are used to create the source tables for the query.)

---

## Joining a Table to Itself

### Problem

You need to produce a subset of information that is based on the relationship between columns in the same table.

### Solution

You can join the table to itself by creating the second version of the table with an alias. Then, you can create a query to compare data from columns in the original table to other columns in the aliased table.

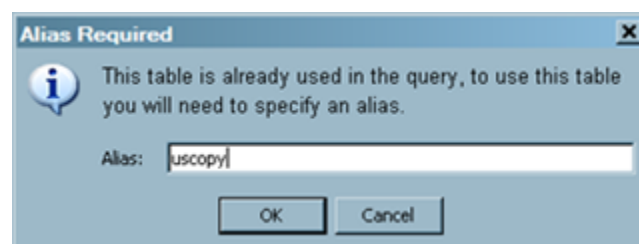
### Tasks

#### *Join the Table to Itself*

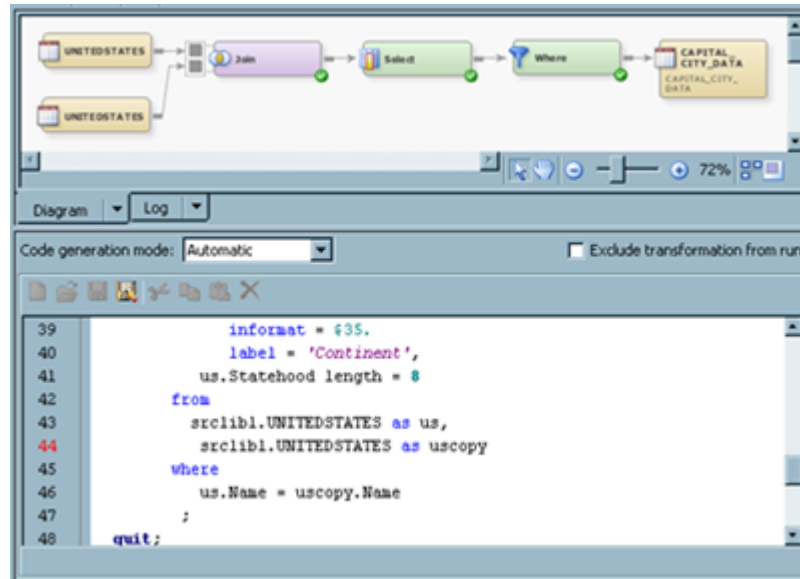
Perform the following steps to join a table to itself and use the resulting hierarchy of tables in a query:

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** window for the SQL Join transformation. Click **Create** in the Navigate pane to access the **Diagram** tab and the SQL Clauses pane.
3. Drop the same table that was used as a source table for the query in the **Diagram** tab. You are prompted to supply an alias for the table because it is already being used as a source table for the query. Enter the alias in the **Alias** field of the properties pane for the table. The dialog box for the alias is shown in the following display.

**Display 17.24** Self-Join Alias Dialog Box



4. Complete any additional configuration needed to finish the query. The following display shows a sample job that includes a table joined to itself.

**Display 17.25** Sample Job with a Table Joined to Itself

The tables in the flow shown on the **Diagram** tab are reflected in the FROM clause that is highlighted on the **Code** tab below it. The query that is 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 that is run in an iterative job. This iterative job contains a control loop in which one or more processes are executed multiple times, so this job needs to be allowed to iteratively run a series of tables in a library through your SQL query. For example, you need to 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 337](#).
- 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 338](#).
- One or more parameters must be set for the parameterized job. See [“Set Parameters for the Job” on page 339](#).

- The parameterized job must be embedded in an iterative job. See [“About Iterative Jobs” on page 333](#).
- The parameters from the parameterized job must be mapped on the **Parameter Mapping** tab of the properties window for the iterative job.
- 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 340](#).

---

## 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, which is known 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 is processed in the SAS SPD Server by 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 in order 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 runs 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 enables 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
;
```

*Note:* 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**

---

## 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 that you choose. 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 enhance performance.

You might often perform lookups between tables in SAS Data Integration Studio. 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 the problems 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 297](#)
- [“Enabling Pass-Through Processing” on page 326](#)
- [“Influencing the Join Algorithm” on page 324](#)
- [“Performing General Data Optimization” on page 323](#)

- [“Understanding Automatic Joins” on page 291](#)
- [“Setting the Implicit Property for a Join” on page 325](#)
- [“Selecting the Join Type” on page 294](#)
- [“Using Properties Window Options to Optimize SQL Processing Performance” on page 328](#)

---

## 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. Perform the following tasks:

- [“Minimize Input/Output \(I/O\) Processing” on page 323](#)
- [“Pre-Sort Data” on page 323](#)

### **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 that is 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 yields the best possible performance for the SQL processing that is included in a SAS Data Integration Studio job. SAS SQL implements several well-known join algorithms: sort-merge, index, and hash.

### Solution

Common techniques 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` option, which adds a trace that indicates which algorithm is used when in the **Log** tab.

You can use the following join types:

- “Sort-Merge Joins” on page 324
- “Index Joins” on page 324
- “Hash Joins” on page 325

### 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 that is selected most often 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 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 that includes 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 **Diagram** tab to access the property in the pop-up menu. The following table depicts the settings that are available for each type of join, along with a sample of the join condition code that is generated for the join type:

**Table 17.5** *Implicit and Explicit Properties for SQL Join Types*

Join Type	Join Condition Code
Inner	<p>Can generate an implicit inner join condition in a WHERE statement near the end of the query:</p> <pre>where     POSTALCODES.Name = UNITEDSTATES.Name</pre> <p>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:</p> <pre>from     srclib.POSTALCODES inner join         srclib.UNITEDSTATES             on                 (                     POSTALCODES.Name = UNITEDSTATES.Name                 )</pre>

Join Type	Join Condition Code
Full	<p>Can generate an explicit join condition in a FROM clause in the SELECT statement:</p> <pre> from   srclib.POSTALCODES full join     srclib.UNITEDSTATES       on         (           POSTALCODES.Name = UNITEDSTATES.Name         ) </pre>
Left	<p>Can generate an explicit join condition in a FROM clause in the SELECT statement:</p> <pre> from   srclib.POSTALCODES left join     srclib.UNITEDSTATES       on         (           POSTALCODES.Name = UNITEDSTATES.Name         ) </pre>
Right	<p>Can generate an explicit join condition in a FROM clause in the SELECT statement:</p> <pre> from   srclib.POSTALCODES right join     srclib.UNITEDSTATES       on         (           POSTALCODES.Name = UNITEDSTATES.Name         ) </pre>
Cross	<p>Can generate an explicit join condition in a FROM clause in the SELECT statement:</p> <pre> from   srclib.POSTALCODES cross join     srclib.UNITEDSTATES </pre>
Union	<p>Can generate an explicit join condition in a FROM clause in the SELECT statement:</p> <pre> from   srclib.POSTALCODES union join     srclib.UNITEDSTATES </pre>

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.

## 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 is not used.

You can use the following types of pass-through processing:

- “Explicit Pass-Through Processing” on page 327
- “Implicit Pass-Through Processing” on page 327

## 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 generates errors. For example, using any of the following in a WHERE clause expression or in a subquery on the WHERE or FROM clauses causes 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 does 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 property 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 does 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, all of the rows in the table are deleted. If the table doesn't exist, it is created.

### Implicit Pass-Through Processing

Even if **Pass Through** is set to **No**, PROC SQL still tries 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 is passed to the database. However, PROC SQL tries to generate SQL that passes 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 approach 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 is not 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 is 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 a good practice to upload the smaller of the tables in the SQL query because this operation can 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 to be 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 that is associated with the host variable is not available to the DBMS that processes the query. The INTO: clause is not supported in the SQL Join transformation.
- **Queries that contain the SAS OUTER UNION operator:** This operator 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 functions vary by database.
- **Specification of ANSIMISS or NOMISS in the join syntax.**

---

## Using Properties Window Options to Optimize SQL Processing Performance

### *Problem*

You want to set specific options in the SQL Properties pane or table properties panes that are located in the Designer window for an SQL Join transformation. These options are intended to improve the performance of SQL processes that are included in a SAS Data Integration Studio job.

## Solution

Use one of the following techniques:

- “Bulk Load Tables” on page 329
- “Optimize the SELECT Statement” on page 329
- “Set Buffering Options” on page 330
- “Use Threaded Reads” on page 330
- “Write User-Written Code” on page 330

## 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, then 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 these 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. For information about using the bulk-load option for each SAS/ACCESS engine, see 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 selects all columns in the order in which they are stored in a table and then runs 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 that is available on some database management systems.

You should experiment with different settings for these options to find optimal performance for your query. These options apply to data sets; therefore, do not specify them unless you know that explicit pass-through or implicit pass-through is not used on that portion of the query because they could actually slow performance. If these options are present in the query at all, they prevent implicit pass-through processing. If these options are present on the part that is being explicitly passed through, a database error occurs 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 option on this target table causes 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, then the database returns an error because it does not recognize this option in the query's FROM clause. To avoid the risk of preventing implicit pass-through, specify these options in the LIBNAME statement, which applies to all tables that use that LIBNAME and anything that accesses those tables. These buffering data set options are great performance boosters if the database records are all 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.

### 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 results from each of these threads, known as a result set, is partitioned across the connections, and rows are passed to SAS simultaneously (in parallel) across the connections. This approach improves performance.

To perform a threaded read, SAS first creates threads, which are standard operating system tasks that are 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 is 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

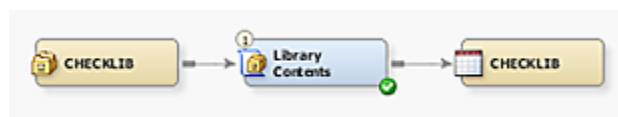
# Working with Iterative Jobs and Parallel Processing

<b>About Iterative Jobs</b> .....	<b>333</b>
<b>Creating and Running an Iterative Job</b> .....	<b>334</b>
Problem .....	334
Solution .....	334
Tasks .....	334
<b>Creating a Parameterized Job</b> .....	<b>337</b>
Problem .....	337
Solution .....	337
Tasks .....	337
<b>Creating a Control Table</b> .....	<b>340</b>
Problem .....	340
Solution .....	340
Tasks .....	340
<b>About Parallel Processing</b> .....	<b>342</b>
<b>Setting Options for Parallel Processing</b> .....	<b>344</b>
Problem .....	344
Solution .....	344
Tasks .....	344

## 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. The circled numbers represent the order in which the transformations are run.

**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.

For an example of how the steps in the iterative process are performed, see [“Creating and Running an Iterative Job” on page 334](#).

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 that are 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 CHECKLIB 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.

## 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.

Perform the following tasks:

- [“Create the Iterative Job ” on page 334](#)
- [“Variation: Add the Library Input and Library Contents Transformation Directly to a Job ” on page 335](#)
- [“Run the Iterative Job and Examine the Results” on page 336](#)

### Tasks

#### Create the Iterative Job

Perform the following steps to create and run the iterative job:

1. Create the control table and the parameterized job that are included in the iterative job. See [“Creating a Control Table” on page 340](#) and [“Creating a Parameterized Job” on page 337](#) for more information.

2. Create an empty job.
3. Select and drag the Loop transformation from the **Control** folder in the Transformations tree. Then, drop it in the empty job on the **Diagram** tab in the Job Editor window.
4. Select and drag the control table from its folder. Then, drop it before the Loop transformation on the **Diagram** tab.
5. Select and drag the parameterized job from its folder. Then, drop it after the Loop transformation on the **Diagram** tab.
6. Select and drag the Loop End transformation from the **Control** folder in the Transformations tree. Then, drop it after the parameterized job on the **Diagram** tab.
7. Drag the control table and connect it to the input port for the Loop transformation.

A sample completed iterative job is shown in the following display.

**Display 18.3** Completed Iterative Job

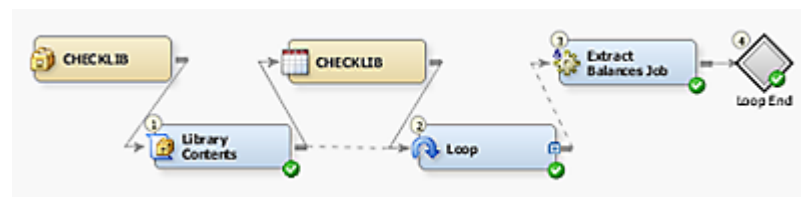


8. Open the **Loop Options** tab in the properties window for the Loop transformation. Select the **Execute iterations in parallel** check box. Also select the **One process for each available CPU node** check box in the **Maximum number of concurrent processes** group box.
9. Open the **Parameter Mapping** tab. Make sure that the appropriate value **Source table** field is mapped to the parameter that is 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.
10. Close the properties window for the Loop transformation.

### ***Variation: Add the Library Input and Library Contents 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 340](#).

### Run the Iterative Job and Examine the Results

After you run the iterative job, you can find output for the completed iterative processing 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. Perform the following steps to run the job, review the status data, and examine the iterative job output:

1. Run the iterative job. The following display shows a successfully completed sample job.

**Display 18.5** Sample Successful Iterative Job

The screenshot shows the SAP Business Objects Designer interface. At the top, a diagram illustrates the job flow: a 'CHECKLIB' transformation leads into a 'Loop' transformation, which then connects to an 'Extract Balances Job' transformation, and finally to a 'Loop End' transformation. Below the diagram, the 'Details' pane is open, showing the 'Status' tab. It indicates the last run was on Feb 22, 2008 at 3:03:23 PM. A table lists the components of the job and their status.

Node	Name	Status	Details
0	Precode	Completed successfully	
1	Loop	Completed successfully	
2	Loop End	Completed successfully	
3	Postcode	Completed successfully	
	Loop Job	Completed successfully	

At the bottom of the Details pane, it states 'Completed successfully'.

2. Right-click the temporary table that is attached to the Loop transformation and click **Open**. A sample View Data window for the status information in the Loop transformation temporary output table is shown in the following example.

**Display 18.6** Loop Transformation Temporary Table

The screenshot shows a 'View Data: Loop (5 rows)' window. It contains a table with 5 rows of iteration data. Each row includes an iteration number, a handle, a machine name, start and end times, and a status.

#	etls_handle...	etls_machi...	etls_startTime	etls_endTime	etls
1	LS2_1	...	February 22, 2008 03:03:15 PM	February 22, 2008 03:03:17 PM	Finished
2	LS2_2	...	February 22, 2008 03:03:15 PM	February 22, 2008 03:03:17 PM	Finished
3	LS2_3	...	February 22, 2008 03:03:18 PM	February 22, 2008 03:03:20 PM	Finished
4	LS2_4	...	February 22, 2008 03:03:19 PM	February 22, 2008 03:03:20 PM	Finished
5	LS2_5	...	February 22, 2008 03:03:20 PM	February 22, 2008 03:03:21 PM	Finished

Each row in this table contains information about an iteration in the job.

3. 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 18.7** View of Target Table Output

#	CHECKING_ID	CHECKING...	CHECKING_APP_MARITAL_ST...	CHECKING_CUF
1	CHK-176	FRIEND	...	74884.42
2	CHK-133	NAPPER	...	52987.30
3	CHK-114	MCCRACKEN	...	64924.80
4	CHK-109	LEASE	...	60290.57
5	CHK-101	HAMILTON	...	99915.28
6	CHK-103	HOWEL	...	67283.00
7	CHK-92	LESLEY	...	92263.59
8	CHK-84	SMYTH	...	51764.39
9	CHK-66	GRESSETT	...	81878.48
10	CHK-28	GLAD	...	70695.78
11	CHK-21	KENNEY	...	93884.94
12	CHK-23	FOWLEY	...	88963.90

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.

## 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.

Perform the following tasks:

- “Create and Populate the Job” on page 337
- “Set Input and Output Parameters” on page 338
- “Set Parameters for the Job” on page 339
- “Complete Parameterized Job Configuration” on page 339

### Tasks

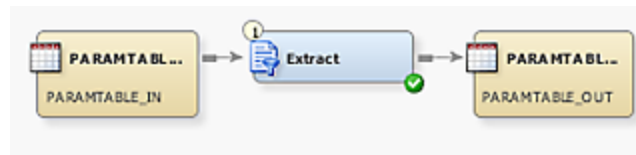
#### Create and Populate the Job

Perform the following steps to create and populate the job:

1. Create and register the input and output tables. The input and output tables must contain exactly the same columns as the tables that are listed in the control table for the loop processing in the iterative job to work properly.
2. Create an empty job.
3. Select and drag the SAS transformation that is used to process the data from the appropriate folder in the Transformations tree. Then, drop it in the empty job on the **Diagram** tab in the Job Editor window. The sample job uses an Extract transformation to extract a subset of the data with a specified marital status from the source tables that are run through the loop.
4. Select and drag the source table from its folder. Then, drop it before the SAS transformation on the **Diagram** tab. You set the input parameter on this table.
5. Drag the cursor from the source table to the input port of the SAS transformation. This action connects the source to the transformation.
6. Because you must have a permanent target table to contain the output parameter that is needed for the loop job to work, right-click the temporary work table attached to the transformation and click **Replace** in the pop-up menu. Then, use the Table Selector window to select the target table for the job. The target table must be registered in SAS Data Integration Studio. (For more information about temporary work tables, see [“Working with Default Temporary Output Tables”](#) on page 126.) You set the output parameter on this table.

A sample completed parameterized job is shown in the following example.

**Display 18.8** Completed Parameterized Job



The input table for the sample job is named PARAMTABLE\_IN. The output table is named PARAMTABLE\_OUT.

### 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 Prompt** to display the New Prompt window. Enter appropriate values in the following fields on the **General** tab:
  - **Name:** a valid macro variable name, such as `mstatus`
  - **Displayed Text:** a display name for the macro variable, such as `Marital Status`.

If you want to enter a default value for the input table, click the **Prompt Type and Values** tab. Then, enter the value in the **Default value** field. The default value in the sample job is `CHECKING_ACCOUNT_DIVORCED`. Because the default prompt type of **Text** is appropriate, you keep the defaulted values in the other fields on the **Prompt Type and Values** tab.

2. Click **OK** to save the parameter and close the New Prompt window.

3. Open the **Physical Storage** tab. Enter an appropriate value in the **Name** field. Create this value by combining an ampersand sign (&) with the value that was entered in the **Macro Variable Name** field in the New Prompt window (for example, **&mstatus**).
4. Click **OK** to save the settings and close the properties window for the input table.
5. Open the **Parameters** tab in the properties window for the output table. Click **New Prompt** to display the New Prompt window. Enter appropriate values in the following fields on the **General** tab:
  - **Name:** a valid macro variable name, such as **mstatus**.
  - **Displayed Text:** a display name for the macro variable, such as **Marital Status Out**.

If you want to enter a default value for the output table, click the **Prompt Type and Values** tab. Then, enter the value in the **Default value** field. The default value in the sample job is **CHECKING\_ACCOUNT\_DIVORCED**. Because the default prompt type of **Text** is appropriate, you keep the defaulted values in the other fields on the **Prompt Type and Values** tab.
6. Click **OK** to save the parameter and close the New Prompt window.
7. Open the **Physical Storage** tab. Enter an appropriate value in the **Name** field. Create this value by combining an ampersand sign with the value that was entered in the **Macro Variable Name** field in the New Prompt window and appending **.OUT** to the combination (for example, **&mstatus.OUT**).
8. Click **OK** to save the settings and close the properties window for the input 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 Parameters** to display the Import Parameters window. Click an appropriate value such as **PARAMTABLE\_IN** in the **Available Parameters** field. Select the parameter that is assigned to the input table and move it to the **Selected Parameters** field. Then, click **OK** to save the setting and 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. For the sample job, the **WHERE** condition is
 

```
CHECKING_APP_MARITAL_STATUS_CD = 'D'
```
2. Open the **Mapping** tab in the properties window for the transformation that is 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.

## 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.

Perform the following tasks:

- “Create and Register the Control Table” on page 340
- “Create and Populate the Job” on page 341
- “Run the Job and Examine the Output” on page 341

### Tasks

#### Create and Register the Control Table

If you have an existing control table, you can use it. If you don't use an existing control table, you can use the Code 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 New Table wizard 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** ⇒ **Code 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;
```

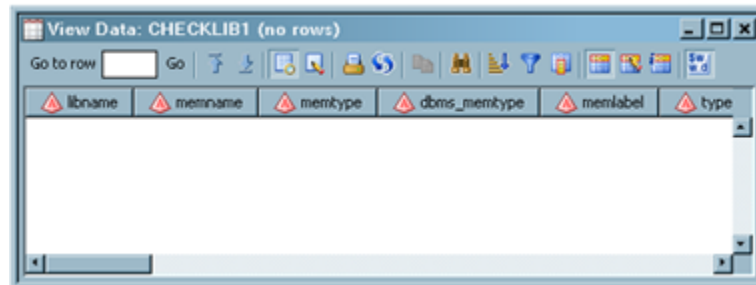
Be sure to check the **Log** tab to verify that the code ran without errors.

3. Register the table that you just created using the Register Tables wizard. This action creates a metadata object for the table.



- (Optional) You can confirm that the empty control table was created in physical storage. Right-click the metadata object for the table and select **Open**. A sample empty control table is shown in the following example.

**Display 18.9** View of Empty Control Table Output



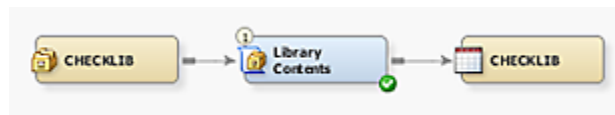
### Create and Populate the Job

Perform the following steps to create and populate the job:

- Create an empty job.
- Select and drag a Library Contents transformation from the **Access** folder in the Transformations tree. Then, drop it in the empty job on the **Diagram** tab in the Job Editor window.
- Select and drag the library that you plan to use to generate the control table from its folder. Then, drop it before the Library Contents transformation on the **Diagram** tab.
- Drag the cursor from the library to the input port of the Library Contents transformation. This action connects the library to the transformation.
- Because you want to have a permanent target table to contain the output for the transformation, right-click the temporary work table that is attached to the transformation and click **Replace** in the pop-up menu. Then, use the Table Selector window to select the target table for the job. The target table must be registered in SAS Data Integration Studio. (For more information about temporary work tables, see [“Working with Default Temporary Output Tables”](#) on page 126.)
- Drag the cursor from the output port of the Library Contents transformation to the target table. This action connects the transformation to the target.
- 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. You can click **Map all columns** to correct any errors.

A sample completed control table job is shown in the following example.

**Display 18.10** Completed Control Table Job

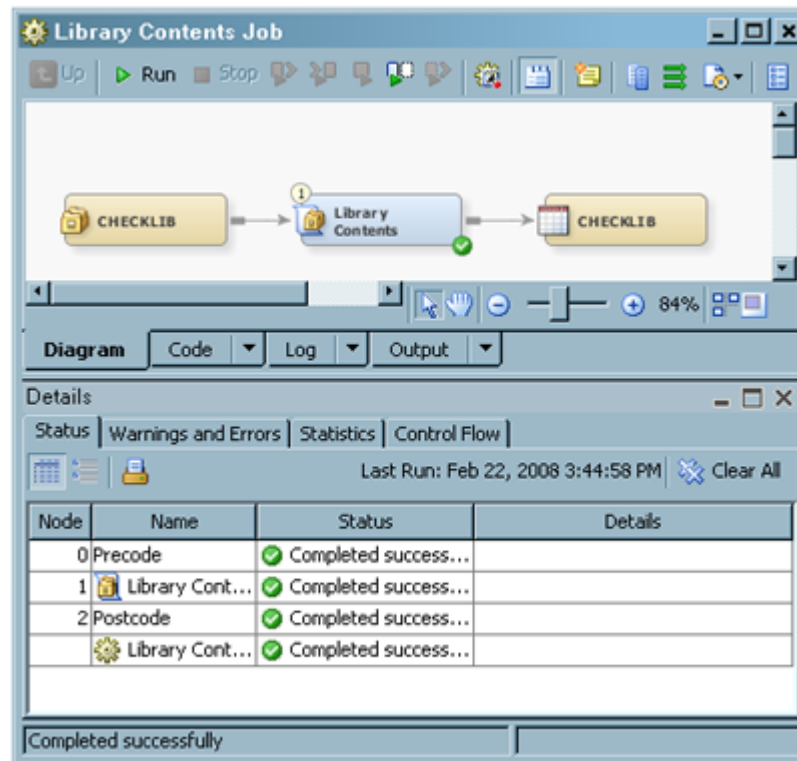


The library for the sample job is named CHECKLIB. The target table is also named CHECKLIB.

### Run the Job and Examine the Output

Perform the following steps to run the control table job and examine its output:

- Run the job. The following display shows a successfully completed sample job.

**Display 18.11** Successful Sample Control Job

- If the job completes without error, right-click the control table icon and click **Open**. The View Data window appears, as shown in the following example.

**Display 18.12** View of Control Table Output

#	libname	memname	memtype	dbms_memtype	memlabel	t
1	CHECKLIB	CHECKINGTRA...	DATA		CheckingTrans...	DATA
2	CHECKLIB	CHECKING_AC...	DATA		...	DATA
3	CHECKLIB	CHECKING_AC...	DATA		CHECKING_AC...	DATA
4	CHECKLIB	CHECKING_AC...	DATA		CHECKING_AC...	DATA
5	CHECKLIB	ETLS_ACTIVITY...	DATA		...	DATA
6	CHECKLIB	ETLS_ACTIVITY...	DATA		...	DATA
7	CHECKLIB	ETLS_ACTIVITY...	DATA		...	DATA
8	CHECKLIB	ETLS_ACTIVITY...	DATA		...	DATA

Note that all of the rows in the table are populated with the name of the control table in the libname column. This name 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.

## 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. These macros enable you to create your own transformations or code in order to use parallel processing.

When you include a Loop transformation in a job, the transformation generates the necessary macros to use 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. Grid options can be enabled for a job even when the grid software has not been configured and licensed. However, SAS Data Integration Studio does not generate grid-enabled code for the job in this case. 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 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	<p>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 <i>SAS Intelligence Platform: Desktop Application Administration Guide</i>.</p> <p>Specify an appropriate SAS®9 Workspace Server in the metadata for the default.</p> <p>Grid software must be licensed.</p> <p>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.</p> <p>Install Platform Computing software to handle workload management for the grid.</p>

*Note:* For additional information about these requirements, see the grid chapter in *SAS Intelligence Platform: Application Server Administration Guide*.

## Setting Options for Parallel Processing

### Problem

You want to use parallel processing and grid processing in SAS Data Integration Studio jobs.

### Solution

If you need to enable parallel or grid processing for all jobs, then set global options on the **Code Generation** tab of the Options window for SAS Data Integration Studio. If you need to enable parallel or grid processing for a single iterative job, then set the options that are 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	Task
Enable parallel processing macros for new jobs	Adds parallel processing macros to the code that is generated for all new jobs.	Select <b>Tools</b> ⇒ <b>Options</b> from the menu bar. Click the <b>Code Generation</b> tab. Specify the desired option.
Various grid computing options	Sets grid computing options for all new jobs.	Select <b>Tools</b> ⇒ <b>Options</b> from the menu bar. Click the <b>Code Generation</b> tab. Specify the desired option.

**Table 18.3** Local Options (affects the current job or transformation)

Option	Purpose	Task
Enable parallel processing macros	<p>When <b>YES</b> is selected, this option adds parallel processing macros to the code that is generated for the current job.</p> <p>Parallel processing macros are always included in the code that is generated for a Loop transformation.</p>	Open the <b>Options</b> tab in the properties window for the job. Select <b>YES</b> or <b>NO</b> in the field for this option.

Option	Purpose	Task
Various grid computing options for the Loop transformation	Sets grid options for the current Loop transformation	Open the <b>Loop Options</b> tab in the properties window for the Loop transformation. Specify the desired option.



## Chapter 19

# Working with Slowly Changing Dimensions

---

<b>About Slowly Changing Dimensions</b> . . . . .	<b>348</b>
Slowly Changing Dimensions Defined . . . . .	348
Types of Slowly Changing Dimensions . . . . .	348
Transformations That Support Slowly Changing Dimensions . . . . .	349
SCD Project Stages . . . . .	349
<b>About Dimension Tables</b> . . . . .	<b>350</b>
About Change Tracking . . . . .	350
About Change Detection and Loading for SCD . . . . .	350
About Generated Keys . . . . .	351
About Cross-Reference Tables . . . . .	352
About Type 1 Updates . . . . .	352
<b>About Fact Tables</b> . . . . .	<b>352</b>
Overview . . . . .	352
About the Loading of Fact Tables with the Lookup Transformation . . . . .	353
<b>Loading a Dimension Table with Type 1 and 2 Updates</b> . . . . .	<b>353</b>
Problem . . . . .	353
Solution . . . . .	353
Tasks . . . . .	353
<b>Loading a Fact Table Using Dimension Table Lookup</b> . . . . .	<b>356</b>
Problem . . . . .	356
Solution . . . . .	356
Tasks . . . . .	357
<b>Loading a Table and Adding a Surrogate Primary Key</b> . . . . .	<b>362</b>
Problem . . . . .	362
Solution . . . . .	362
Tasks . . . . .	362
<b>Tracking Changes in Source Datetime Values</b> . . . . .	<b>365</b>
Problem . . . . .	365
Solution . . . . .	365
Tasks . . . . .	365
<b>Closing Out Rows in Datetime Change Tracking</b> . . . . .	<b>367</b>
Problem . . . . .	367
Solution . . . . .	367

## About Slowly Changing Dimensions

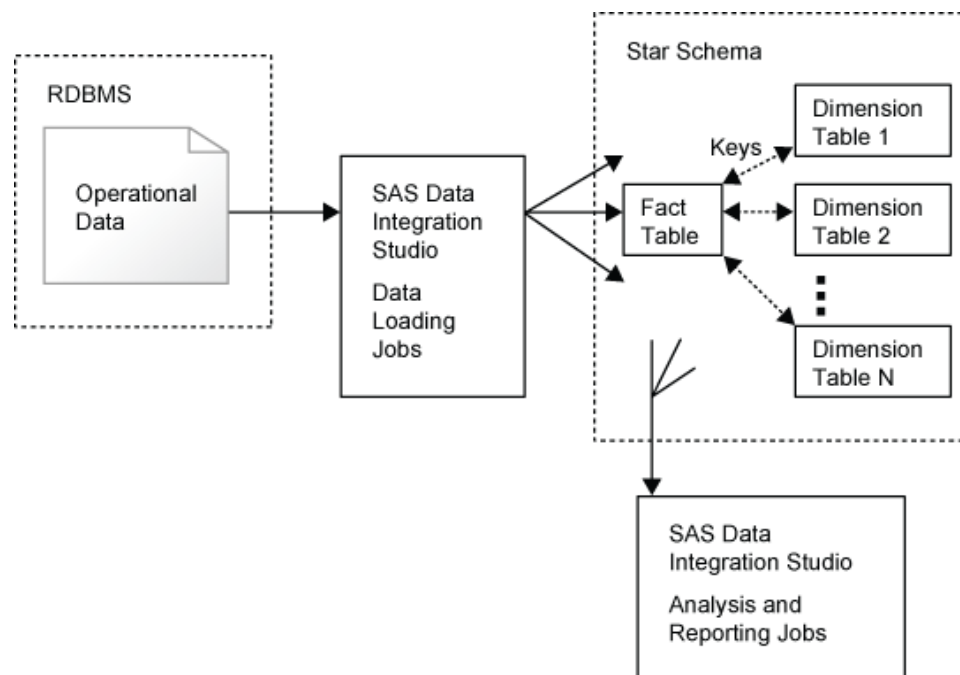
### Slowly Changing Dimensions Defined

Slowly changing dimensions (SCD) is the name of a process that loads data into dimension tables. The dimension tables are structured so that they retain a history of changes to their data. This record of data changes provides a basis for analysis.

As shown in the following diagram, dimension tables combine with fact tables to form star schemas. Fact tables store numeric events. Dimension tables store the detail data that describes the events. Key columns in the tables connect events to details. For example, a star schema might store product sales numbers in a fact table, and use dimension tables to store information about customers, suppliers, and retail locations.

You can use SAS Data Integration Studio to load data into star schemas and analyze data to extract knowledge from the star schema.

**Figure 19.1** The Star Schema and SAS Data Integration Studio



In SAS Data Integration Studio, the process of loading dimension tables takes place in the SCD Type 2 Loader transformation. Fact tables are loaded with the Lookup transformation.

### Types of Slowly Changing Dimensions

The three types of slowly changing dimensions are defined as follows:

Type 1 SCD: no history of data changes

overwrites specified columns in dimension tables without retaining a history of changes. Type 1 SCD is useful for maintaining less-significant columns that are not used in historical analysis. In SAS Data Integration Studio, the SCD Type 2 Loader transformation allows you to combine Type 1 and Type 2 updates in a single operation.



**Type 2 SCD: full history**

maintains multiple records for each individual in the dimension table. The latest entry is the current entry for that member. Other rows comprise the historical record of data changes. New entries create new current rows. This comprehensive record of data changes is the primary purpose of the SCD Type 2 Loader transformation.

**Type 3 SCD: limited history**

maintains a limited history of changes using multiple columns for selected variables. For example, a Type 3 dimension table containing customer information has columns named New Postal Code, Old Postal Code, and Oldest Postal Code. Data is moved from column to column during the loading process. Type 3 SCD has less analytical value than Type 2 SCD.

## ***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 exception handling. The lookup process accesses dimension tables by using hash objects for optimal performance.

**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.

## ***SCD Project Stages***

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 Dimension Tables

### About Change Tracking

Dimension tables that are loaded with the SCD Type 2 Loader consist of a primary key column, a business key column, 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. You can combine tracking methods as needed to optimize your analyses. Using a current-row indicator column improves the performance of the SCD Type 2 Loader.

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. The begin datetime for the new current row is one second greater than the end datetime of the former current row. The end value for the current row is a placeholder future date.

**Figure 19.2** Structure of an SCD Dimension Table

	Business Key	Begin Date and Time	End Date and Time	Version	Current Row	Generated Key	Detail Data
Current Row	2138	27JUL200800:01:09	01JAN259900:00:00	3	1	25	
Closed-out Row	2138	15MAY200800:03:23	27JUL200800:01:08	2	0	18	
Closed-out Row	2138	22FEB200800:02:17	15MAY200800:03:22	1	0	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 Loading for SCD

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 as the new current member for that business key. The current member contains the latest information. The loading process moves 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 as the new current row for that business key. The loading process moves on to the next source row.
3. If business keys match and data differences are detected in the columns specified for Type 2 SCD, 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 specified, depending on the selected method of change tracking. For Type 1 updates, if changes are detected in the Type 1 columns, the source data overwrites the target data in the current row, even if data differences are not detected in the Type 2 columns.

## About Generated Keys

The SCD Type 2 Loader enables you to generate key values when you load a dimension table. The generated values are frequently used as primary keys, because the business key from the source table identifies the member, not the unique row in the dimension table.

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 for each new row. You can also use an expression to generate key values in other increments. To specify a unique starting point for the keys that are generated in each load, you can specify a lookup column. The initial key value is the highest value in the lookup column.

*Note:* When loading a fact table instead of a dimension table, you can generate simple surrogate keys using the Lookup transformation.

In addition to surrogate keys, you can also generate retained keys. Retained keys provide a primary key value that consists of two columns, the begin datetime change tracking column and a numeric column that receives generated values. The combination of the two columns uniquely identifies each row in the table.

The generated value is retained because a single generated value is applied to all of the rows that apply to a given member. When a new row is added to an existing member, it receives the same generated value as the other rows that apply to that member.

As with surrogate keys, you can generate retained key values using expressions and lookup columns.

In order to generate unique retained keys, begin and end datetime change tracking is required.

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 that consists of one column. Retained keys should receive a complex index that includes the generated key column and the beginning datetime column.

To create an index, open the Properties dialog box for the table and use the **Index** and **Keys** tabs.

## 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 is 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.

To increase performance in large jobs, enable change tracking by current row indicator. This method of change tracking can be combined with the other change tracking methods (begin and end datetime and version number). The current row indicator speeds up the process of creating or updating the digest file. The performance improvement is provided by a `WHERE` clause that efficiently separates current rows from closed-out rows.

Cross-reference tables are identified on the **Options** tabs of the following transformations: SCD Type 2 Loader and Key Effective Date, in the field **Cross-Reference Table Name**.

## About Type 1 Updates

Type 1 updates are defined as overwrites of existing data in specified columns. When you run a Type 1 update with the SCD Type 2 Loader transformation, digest values containing the Type 1 columns are created for the source and target. The digest values are then compared to determine the target rows that need to be updated. When the rows are updated, the number of writes is optimized.

You can combine Type 2 and Type 1 updates in the same job. Use Type 2 updates to maintain a history of changes for important columns. Use Type 1 updates to maintain accurate and complete information in your dimension table, without generating new target rows for each change.

---

## About Fact Tables

### Overview

Fact tables are combined with dimension tables to make up star schemas. Fact tables describe events using numeric data. Dimension tables provide detail data that describe the events. Examples of factual events include the sale of an item or a transaction in a bank account. Each such event is represented by a single row in a fact table.

The columns in a fact table consist of one or more numeric columns that relate to an event and a series of foreign key columns that connect the event to the detail data in the dimension tables.

### About the Loading of Fact Tables with the Lookup Transformation

To load data into a fact table, use the Lookup transformation in a SAS Data Integration Studio job. The Lookup transformation generates primary key values, loads numeric fact data from a source table, and loads foreign keys from dimension tables using a lookup process.

The lookup process runs separately for each dimension table that contributes foreign keys. The process compares business key values between the source table and a dimension table. If a match is found, an expression (a WHERE clause) is evaluated to identify the specific dimension table row in that business key. In general, the values that are loaded from the dimension table are the primary key columns. Loading these foreign keys into the fact table allows each event to contain references to all of the detail data that describes that event.

If no match is found in a dimension table, or if a value is missing, then the numeric data in the source row is not loaded into the fact table and the exception condition is processed by the Lookup transformation. Each exception condition triggers one or more available actions, including the termination of the job, the loading of source data into an error table, and the loading of information into an exception table.

---

## Loading a Dimension Table with Type 1 and 2 Updates

### Problem

You want to load a dimension table using type 1 updates (overwrites) in certain columns and type 2 updates (track changes) in other columns. You need to generate a primary key for each target row and optimize performance for large source tables.

### Solution

You can create a job that includes the SCD Type 2 Loader transformation. You can load Type 1 and Type 2 changes in a single transformation. To optimize performance, you can add a current-row indicator that speeds up the creation of the cross-reference table that is used for change detection.

The sample job includes the following tasks:

- [“Create and Populate the Job” on page 353](#)
- [“Configure the SCD Type 2 Loader” on page 355](#)
- [“Run the Job and View the Output” on page 355](#)

### Tasks

#### Create and Populate the Job

Perform the following steps to create and populate the job:

1. Create an empty SAS Data Integration Studio job.
2. In the Transformations tree, in the **Data** folder, drag the SCD Type 2 Loader transformation into the empty job on the **Diagram** tab.

3. Select and drag the source table from its folder and drop it before the SCD Type 2 Loader transformation on the **Diagram** tab. In this sample job, the source contains information on customers.
4. Drag the cursor from the source table to the input port of the SCD Type 2 Loader transformation. This action connects the source to the transformation.
5. Create a new target table using the New Table Wizard. The sample job uses the same columns as the source, and adds columns for change tracking, performance enhancement, and a generated key. The new columns are defined as follows:

**VALID\_FROM\_DTTM**

receives begin datetime values.

**VALID\_TO\_DTTM**

receives end datetime values.

**CURRENT\_ROW**

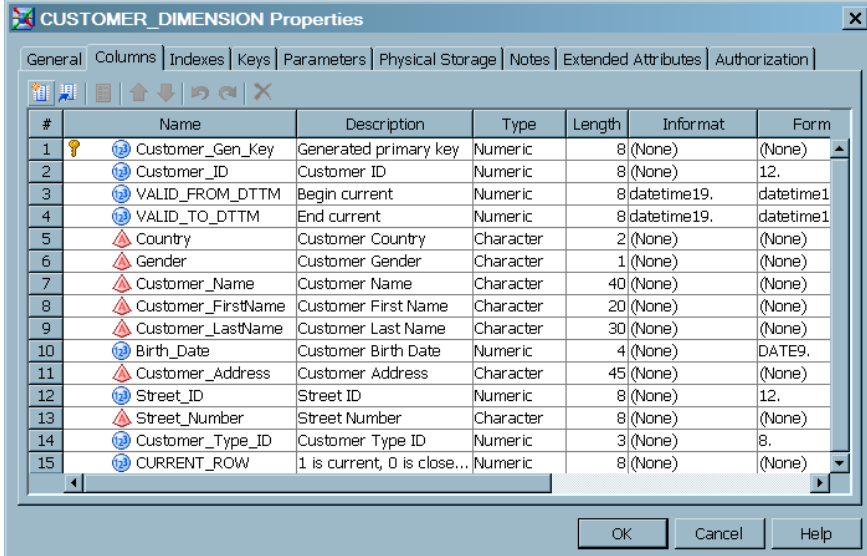
receives 1s in current rows and zeros in closed-out rows. Adding this column improves performance in loads that involve large amounts of data. The current row indicator speeds up the process of creating and updating the cross-reference table.

**CUSTOMER\_DIM\_ID**

receives the generated key values.

The following display shows the column properties for the new target table:

**Display 19.1** Target Column Properties



#	Name	Description	Type	Length	Informat	Form
1	Customer_Gen_Key	Generated primary key	Numeric	8 (None)	(None)	(None)
2	Customer_ID	Customer ID	Numeric	8 (None)	12.	
3	VALID_FROM_DTTM	Begin current	Numeric	8 datetime19.	datetime1	
4	VALID_TO_DTTM	End current	Numeric	8 datetime19.	datetime1	
5	Country	Customer Country	Character	2 (None)	(None)	(None)
6	Gender	Customer Gender	Character	1 (None)	(None)	(None)
7	Customer_Name	Customer Name	Character	40 (None)	(None)	(None)
8	Customer_FirstName	Customer First Name	Character	20 (None)	(None)	(None)
9	Customer_LastName	Customer Last Name	Character	30 (None)	(None)	(None)
10	Birth_Date	Customer Birth Date	Numeric	4 (None)	DATE9.	
11	Customer_Address	Customer Address	Character	45 (None)	(None)	(None)
12	Street_ID	Street ID	Numeric	8 (None)	12.	
13	Street_Number	Street Number	Character	8 (None)	(None)	(None)
14	Customer_Type_ID	Customer Type ID	Numeric	3 (None)	8.	
15	CURRENT_ROW	1 is current, 0 is close...	Numeric	8 (None)	(None)	(None)

6. Drag the target table from its folder and drop it after the SCD Type 2 Loader transformation on the **Diagram** tab.
7. Drag the cursor from the output port of the SCD Type 2 Loader transformation to the target table. This action connects the transformation to the target. The following display depicts the process flow in the sample job.

**Display 19.2** Sample SCD Type 2 Loader Process Flow Diagram



### ***Configure the SCD Type 2 Loader***

Perform the following steps to configure the SCD Type 2 Loader:

1. Open the properties window of the SCD Type 2 Loader and select the **Change Tracking** tab. Note that datetime change tracking is enabled by default, with datetime values delivered to the columns `VALID_FROM_DTTM` and `VALID_TO_DTTM`.
2. Select **Use current indicator**, and then click the down arrow in **Current indicator column**. Select the `CURRENT_ROW` column.
3. Open the **Business Key** tab and specify the source columns that comprise the business key. During change detection, the business key columns are compared between each incoming source row and the entire target. If the business keys match between the source and target, then data values are compared to detect changes. Frequently, the business key is the primary key in the source. For the purposes of this example, click **New** and select `Customer_ID`.
4. Open the **Detect Changes** tab and specify the columns that are tracked for Type 2 updates. The number and length of these columns affects the run-time performance of the job. In the sample job, select `Street_ID` and `Customer_Type_ID`, and then click the right arrow.
5. Open the **Type 1 Columns** tab and specify the columns that are updated in the most current rows of their respective business keys, without affecting the begin and end datetime values. Select `Customer_Lastname` and `Customer_Address`, and then click the right arrow.
6. Open the **Generated Key** tab and specify the numeric column that receive the generated key value. Click the down arrow in the **Column** field and specify `CUSTOMER_DIM_ID`. When the job runs, unique identifiers are added to this column for each row in the table.
7. Click **OK** to save changes and close the properties window.

### ***Run the Job and View the Output***

Perform the following steps to run the job and view the output:

1. Right-click on an empty area of the job, and click **Run** in the pop-up menu. SAS Data Integration Studio generates code for the job and submits it to the SAS Application Server for execution.
2. If error messages are displayed on the **Status** tab, please read and respond to the messages as needed.
3. After the completion of the job, right-click the target and select **Open** to view the generated surrogate key values. The following display depicts the target table data for the sample job.

**Display 19.3** Key Columns and Change Tracking Columns in the Sample Target Table

#	Customer_Gen_Key	Customer_ID	VALID_FROM_DTTM	VALID_TO_DTTM	Country	
1	1	1	27MAY2008:10:50:50	01JAN5999:00:00:00	FR	M
2	2	2	27MAY2008:10:50:50	01JAN5999:00:00:00	ES	F
3	3	3	27MAY2008:10:50:50	01JAN5999:00:00:00	IT	M
4	4	4	27MAY2008:10:50:50	01JAN5999:00:00:00	US	M
5	5	5	27MAY2008:10:50:50	01JAN5999:00:00:00	US	F
6	6	6	27MAY2008:10:50:50	01JAN5999:00:00:00	BE	M
7	7	7	27MAY2008:10:50:50	01JAN5999:00:00:00	ES	F
8	8	8	27MAY2008:10:50:50	27MAY2008:10:50:50	FI	M
9	9	8	27MAY2008:10:50:51	27MAY2008:10:50:51	FI	M
10	10	8	27MAY2008:10:50:52	27MAY2008:10:50:52	FI	M
11	11	8	27MAY2008:10:50:53	01JAN5999:00:00:00	FI	M
12	12	9	27MAY2008:10:50:50	27MAY2008:10:50:50	DE	F
13	13	9	27MAY2008:10:50:51	27MAY2008:10:50:51	DE	F
14	14	9	27MAY2008:10:50:52	27MAY2008:10:50:52	DE	F

## Loading a Fact Table Using Dimension Table Lookup

### Problem

You want to load numeric source data into a fact table and add foreign keys from a dimension table.

### Solution

You can create a job that uses the Lookup transformation, which loads fact data from a source table and uses a lookup process to load foreign keys from the dimension table.

The lookup process compares the business key in each source row to the business keys in the dimension table. When the business keys match, the foreign key from the dimension table is loaded into the fact table target.

This sample job assumes that you have already loaded data into your dimension table before you run the job that loads your fact table. Loading the dimension table first ensures that new foreign keys are available in the dimension table.

The sample job includes the following tasks:

- [Create and Populate the Job on page 357](#)
- [Map Source Columns Into the Target on page 357](#)
- [Map Key Columns Between the Source and Lookup Tables on page 358](#)
- [Map Lookup Columns Into the Target on page 359](#)
- [Create Error and Exception Tables on page 359](#)
- [Configure Exception Handling on page 360](#)
- [Run the Job and View the Output on page 360](#)



## Tasks

### Create and Populate the Job

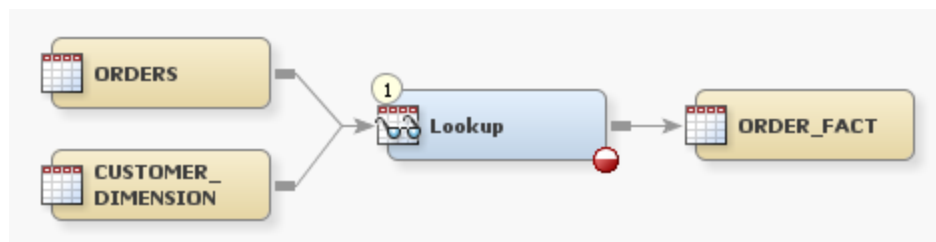
Perform the following steps to load a fact table:

1. Create an empty SAS Data Integration Studio job.
2. In the Transformations tree, in the **Data** folder, drag the Lookup transformation into the empty job in the **Diagram** tab.
3. Select and drag the source table containing numeric fact table data into the source table location on the **Diagram** tab.
4. Drag the cursor from the source table to the input port of the Lookup transformation. This action connects the source to the transformation.
5. Select and drag the lookup table that contains detail data into the **Diagram** tab, into a location that is near the source table.
6. Drag the cursor from the lookup table to the input port of the Lookup transformation. This action connects the lookup table to the transformation.

*Note:* To add more lookup tables, right-click the Lookup transformation and click **Add Input**.

7. Because you want to store the output of the transformation in a permanent target table, right-click the temporary work table that is attached to the transformation and select **Replace**. Then, use the Table Selector window to select the target table for the job. The target table must be registered in SAS Data Integration Studio. (For more information about temporary work tables, see [“Working with Default Temporary Output Tables”](#) on page 126.)
8. Select and drag the target table into the target table location on the **Diagram** tab. The target table has columns for data that is loaded from the source and from the lookup table.
9. Drag the cursor from an output port of the Surrogate Key Generator transformation to the target table. This action connects the transformation to the target. The following example shows the sample process flow.

**Display 19.4** Sample Lookup Process Flow Diagram




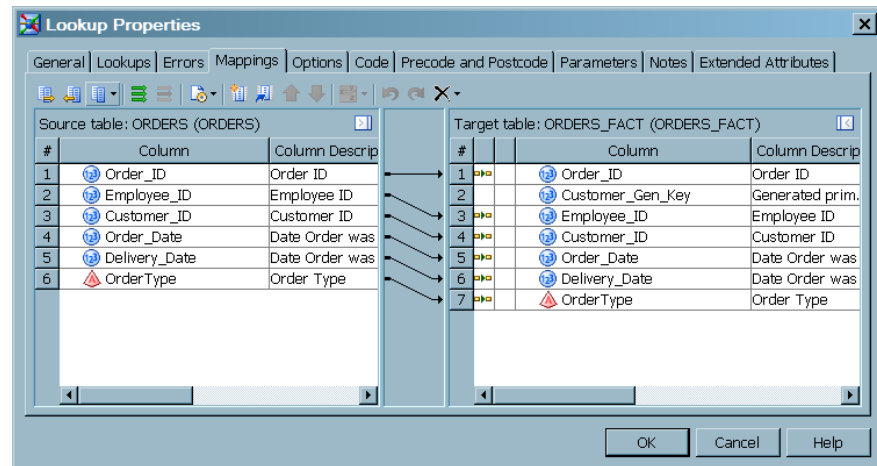
*Note:* In the display, the red icon indicates that the new Lookup transformation requires source column mappings. Click the red icon to display information about errors.

### Map Source Columns Into the Target

Perform the following steps to map fact table columns from the source into the target:

1. In the properties window of the Lookup transformation, open the **Mappings** tab. Use this tab to map the columns directly from the source table to the target table, without the involvement of a lookup table.

- In this sample job, map all source columns to the target by clicking the **Map all columns** icon (  ). The following display depicts the mappings between the source and the target:

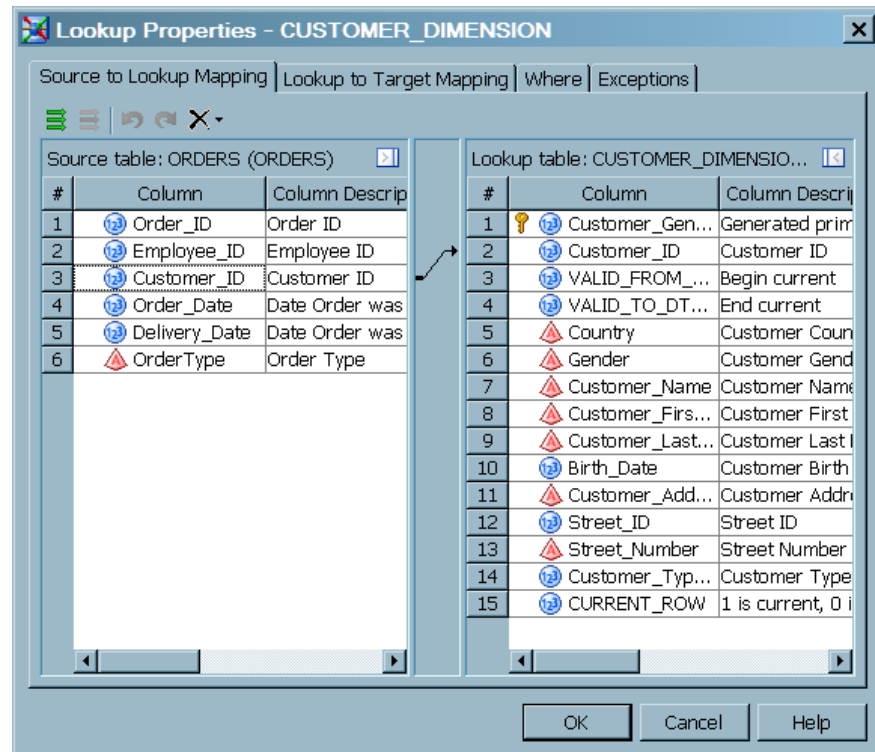
**Display 19.5** Mappings Between Source and Target

### Map Key Columns between the Source and Lookup Tables

Perform the following steps to define the conditions under which values from a lookup table are loaded into the target.

- Select the **Lookups** tab.
- Select the lookup table and click **Lookup Properties**.
- Use the **Source to Lookup Mapping** tab to specify the source and lookup columns that are compared at runtime. If values match, then the lookup value is added to the target. If a match is not found, then an exception condition exists.

In the sample job, the business key in the source is compared to the business key in the lookup table, which in this case is a dimension table that contains customer information. To map the columns, click the **Customer\_ID** column in the **Source Table** list. Then right-click the **Customer\_ID** column in the **Lookup Table** list, and select **Map Selected**. A mapping arrow appears between the two columns. The following display depicts the completed **Source to Lookup Mapping** tab.

**Display 19.6** Source to Lookup Column Mapping

- 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 351.

### Map Lookup Columns Into the Target

Perform the following steps to map lookup columns into the target. Values are loaded when keys match between the source table and lookup table. In the sample job, the target receives lookup table key values. In the target, the key values connect the factual events (orders) to detail data (customer information).

- Open the **Lookup to Target Mapping** tab, and select the **Customer\_Gen\_Key** column.
- Right-click the **Customer\_Gen\_Key** column and select **Map Selected**. A mapping arrow appears between the two columns.

### Create Error and Exception Tables

You can create error and exception tables that receive selected data in response to selected conditions. You configure the error and exception conditions later in this sample job. Perform the following steps to create the error and exception tables:

- Open the properties window of the Lookup transformation and select the **Errors** tab.
- Click **Create error table** and then click **Choose columns**.
- In the Choose Error Table Columns window, note that all source columns are selected to appear in the error table. Click **OK** to close the window.
- On the **Errors** tab, click **Create Exception Table** and click **Choose columns**.

5. In the Choose Exception Table Columns window, note that the exception table columns include the source row number, the lookup table name, the exception condition, and the exception action. Click **OK** to close the window.

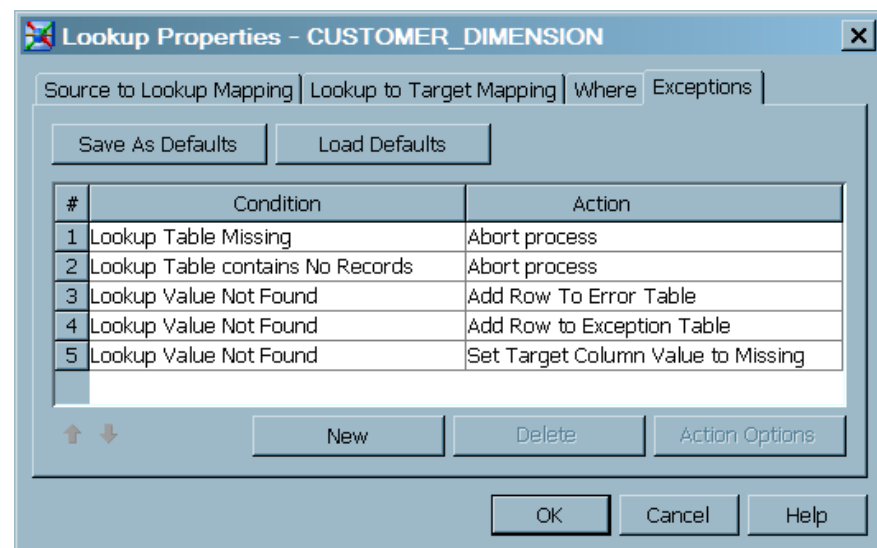
### Configure Exception Handling

If you create an error table and an exception table, the Lookup transformation will, by default, send non-matching source rows to the error table and send exception messages to the exception table. This sample job examines and accepts the default conditions and actions for exception handling.

Perform the following steps to view the default exception handling:

1. In the properties window of the Lookup transformation, select the **Lookups** tab.
2. In the **Lookups** tab, select the lookup table and then select **Lookup Properties**.
3. In the Lookup Properties window, open the **Exceptions** tab. The following display depicts the default configuration for exception handling.

**Display 19.7** Default Exception Handling



In this sample job, if the Customer\_ID column in a source row does not match a Customer\_ID value in the target, then the error and exception tables are updated and the lookup value (Customer\_Gen\_Key) is set to missing for that row in the target.

4. Click **OK** to store your entries and close the properties window of the Lookup transformation.

### Run the Job and View the Output

Perform the following steps to run the job and view the output:

1. Right-click on an empty area of the job, and click **Run** in the pop-up menu. SAS Data Integration Studio generates code for the job and submits it to the SAS Application Server for execution.
2. If error messages are displayed on the **Status** tab, read and respond to the messages as needed.
3. After the completion of the job, right-click the target and select **Open** to view the values that were loaded from the source and lookup tables. The following display depicts the target table data for the sample job.

**Display 19.8** Target Table Data

#	Order_Key	Employee_ID	Customer_ID	Order_Date	Delivery_Date	OrderType
615383		99999999	94252	08JUN2002	10JUN2002	Datalog
615384		99999999	94252	08JUN2002	10JUN2002	Datalog
615385		120391	94252	07JUL2002	07JUL2002	Retail
615386		120391	94252	07JUL2002	07JUL2002	Retail
615387		120357	94252	26MAY2002	26MAY2002	Retail
615388		99999999	94253	29APR2001	03MAY2001	Internet
615389		99999999	94253	22AUG2001	26AUG2001	Internet
615390		99999999	94253	18JAN2002	22JAN2002	Datalog
615391		99999999	94254	23DEC2000	28DEC2000	Datalog
615392		99999999	94254	23DEC2000	28DEC2000	Datalog
615393		99999999	94254	06DEC2000	11DEC2000	Datalog
615394		99999999	94254	06DEC2000	11DEC2000	Datalog
615395		99999999	94254	25OCT2000	30OCT2000	Datalog
615396		99999999	94254	20MAR2000	25MAR2000	Datalog

- To view the contents of the error table, position the cursor in the job, over the Lookup transformation. When the error and exception tables appear, move the cursor over the error table, right-click, and select **Open**. The following display depicts the error table data for the sample job.

**Display 19.9** Error Table Data

#	etls_source_row	Order_ID	Employee_ID	Customer_ID	Order_Date	Delivery_Date
1	108		121073	10	03FEB2001	03FEB2001
2	109		121044	10	12MAR2001	12MAR2001
3	110		121036	10	20OCT2001	20OCT2001
4	111		121036	10	20OCT2001	20OCT2001
5	112		121036	10	20OCT2001	20OCT2001
6	113		121063	10	07OCT2001	07OCT2001
7	114		121039	10	18SEP2001	18SEP2001
8	115		121040	10	18APR2002	18APR2002
9	116		121040	10	18APR2002	18APR2002

- To view the contents of the exception table, position the cursor over the Lookup transformation. When the error and exception tables appear in the job, slide the cursor over the exception table, right-click, and select **Open**. The following display depicts the exception table data for the sample job.

**Display 19.10** Exception Table Data

#	etls_source_row	etls_lookup_table	etls_exception_cond	etls_exception_action
1	108	tglib1.CUSTOMER_DI...	Lookup Value Not Found	Add Row to Exception Ta...
2	109	tglib1.CUSTOMER_DI...	Lookup Value Not Found	Add Row to Exception Ta...
3	110	tglib1.CUSTOMER_DI...	Lookup Value Not Found	Add Row to Exception Ta...
4	111	tglib1.CUSTOMER_DI...	Lookup Value Not Found	Add Row to Exception Ta...
5	112	tglib1.CUSTOMER_DI...	Lookup Value Not Found	Add Row to Exception Ta...
6	113	tglib1.CUSTOMER_DI...	Lookup Value Not Found	Add Row to Exception Ta...
7	114	tglib1.CUSTOMER_DI...	Lookup Value Not Found	Add Row to Exception Ta...
8	115	tglib1.CUSTOMER_DI...	Lookup Value Not Found	Add Row to Exception Ta...
9	116	tglib1.CUSTOMER_DI...	Lookup Value Not Found	Add Row to Exception Ta...

---

## Loading a Table and Adding a Surrogate Primary Key

### Problem

You want to create a job that loads source data into a target and adds a primary key column. The added key column is known as a surrogate key. The surrogate key in the target replaces the primary key that is loaded into the target from the source. The surrogate key is required because the target contains multiple instances of the primary key in the source.

### Solution

You can create a job that includes the Surrogate Key Generator transformation. This transformation is more efficient than the SCD Type 2 Loader because you are not tracking data changes in the target.

The sample job includes the following tasks:

- “Create and Populate the Job” on page 362
- “Add the Surrogate Key Column to the Target” on page 363
- “Identify Tables and Columns in the Transformation” on page 363
- “Run the Job and View the Output” on page 364

### Tasks

#### Create and Populate the Job

Perform the following steps to create and populate the job:

1. Create an empty SAS Data Integration Studio job.
2. In the Transformations tree, in the **Data** folder, drag the Surrogate Key Generator transformation into the empty job on the **Diagram** tab.
3. Select and drag the source table from its folder and drop it before the Surrogate Key Generator transformation on the **Diagram** tab.
4. Drag the cursor from the source table to the input port of the Surrogate Key Generator transformation. This action connects the source to the transformation.
5. Because you want to store the output of the transformation in a permanent target table, right-click the temporary work table attached to the transformation and select **Replace**. Then, use the Table Selector window to select the target table for the job. The target table must be registered in SAS Data Integration Studio. (For more information about temporary work tables, see “Working with Default Temporary Output Tables” on page 126.)
6. Drag the target table from its folder and drop it after the Surrogate Key Generator transformation on the **Diagram** tab.
7. Drag the cursor from an output port of the Surrogate Key Generator transformation to the target table. This action connects the transformation to the target. The following example shows the sample process flow.

**Display 19.11** Sample Surrogate Key Process Flow Diagram

### Add the Surrogate Key Column to the Target

Perform the following steps to add a new column to the target for the generated key values:

1. Open the properties window of the target and select the **Columns** tab.
2. On the **Columns** tab, click the **New column** icon. A new column appears at the bottom of the list.
3. Type the name of the new column. This sample uses the name **CUSTOMER\_GEN\_KEY**.
4. In the **Type** column, change the type of the new column to **Numeric**.
5. To reposition the surrogate key column, select its column number in the list and drag the column up to position 1. The following display depicts the completed **Columns** tab for the sample job.

**Display 19.12** Completed Columns Tab for Sample Job

#	Name	Description	Type	Length	Informat
1	CUSTOMER_GEN_KEY		Numeric	8 (None)	(None)
2	CUSTOMER_ID	CUSTOMER_ID	Numeric	8 (None)	(None)
3	VALID_FROM_DTTM		Numeric	8 (None)	DATE
4	VALID_TO_DTTM		Numeric	8 (None)	DATE
5	SOCIAL_SECURITY_...	SOCIAL_SECURITY_NBR	Numeric	8 (None)	(None)

6. Click **OK** to save your changes and close the properties window.

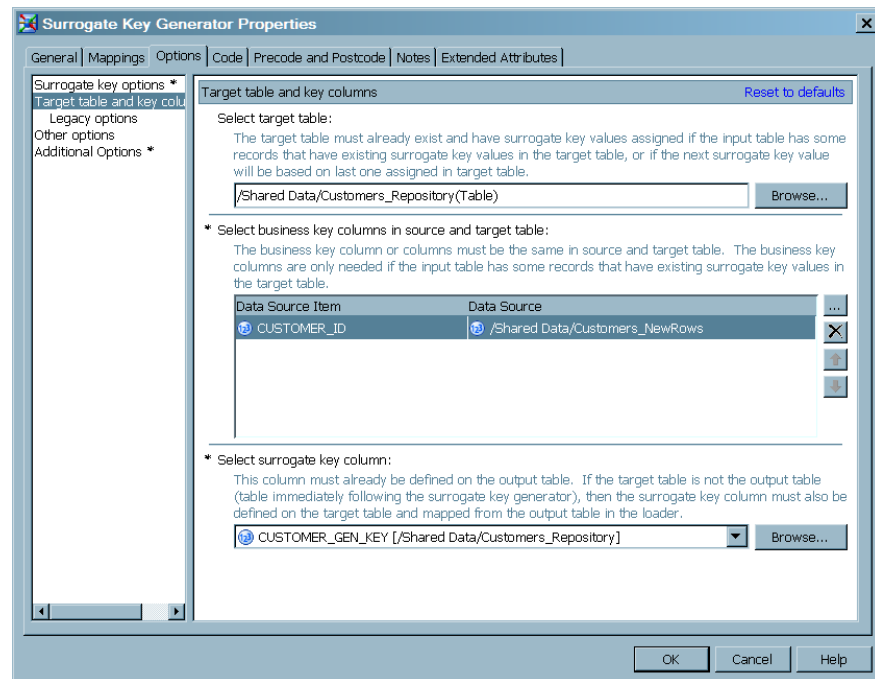
### Identify Tables and Columns in the Transformation

The goal of this section is to configure the Surrogate Key Generator transformation. In this sample job, the surrogate key is generated using the default settings. By default, the transformation generates key values based on the largest value in the key column. Remaining configuration steps identify the target table and the key column in the transformation's **Options** tab. the option values that determine the method of surrogate key generation.

Perform the following steps to configure the Surrogate Key Generator transformation:

1. Open the properties window of the transformation, and then select the **Options** tab. On the **Options** tab, select **Target table and key columns**.
2. Specify the name of the target table in **Select Target Table**.
3. Specify the business key column or columns by selecting from the list of columns under **Select business key columns in source and target table**. The business key columns are the primary key columns in the source.

- Specify the target column that receives the surrogate key values. Select the down arrow under **Select surrogate key column**, and click the target column. The following display depicts the completed **Options** tab in the sample job.

**Display 19.13** Completed Options Tab for Sample Job

- Click **OK** to save the option specifications and close the properties window.

### Run the Job and View the Output

Perform the following steps to run the job and view the output:

- Right-click on an empty area of the job, and click **Run** in the pop-up menu. SAS Data Integration Studio generates code for the job and submits it to the SAS Application Server for execution.
- If error messages are displayed on the **Status** tab, read and respond to the messages as needed.
- After the completion of the job, right-click the target and select **Open** to view the generated surrogate key values. The following display depicts the target table data for the sample job.

**Display 19.14** Generated Key Values in the Sample Target Table

The screenshot shows a window titled 'View Data: Customer\_Repository (100...)'. It displays a table with columns: '#', 'CUSTOMER\_GEN\_KEY', 'CUSTOMER\_ID', and a date column. The data is as follows:

#	CUSTOMER_GEN_KEY	CUSTOMER_ID	
1	2	7369	14DEC
2	3	7499	14DEC
3	4	7505	14DEC
4	5	7506	14DEC
5	6	7507	14DEC
6	7	7521	14DEC



## Tracking Changes in Source Datetime Values

### Problem

You want to track changes to primary key values using begin and end datetime values.

### Solution

You can create a job that uses a Key Effective Date transformation.

The sample job includes the following tasks:

- “Create and Populate the Job” on page 365
- “Identify Source Columns” on page 365
- “Run the Job and View the Output” on page 366

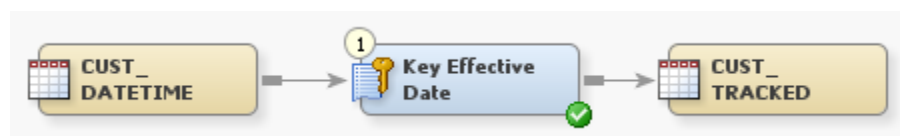
### Tasks

#### Create and Populate the Job

Perform the following steps to create and populate a new job:

1. Create an empty SAS Data Integration Studio job.
2. In the Transformations tree, in the **Data** folder, drag the **Key Effective Date** transformation into the empty job on the **Diagram** tab.
3. Select and drag the source table into the source table location in the **Diagram** tab. In this sample job, the source table contains customer information.
4. Drag the cursor from the source table to the input port of the Key Effective Date transformation. This action connects the source to the transformation.
5. Select and drag the target table into the target table location in the **Diagram** tab. The target contains the same columns as the source.
6. Drag the cursor from an output port of the Key Effective Date transformation to the target table. This action connects the transformation to the target. The following example shows the sample process flow.

**Display 19.15** Sample Key Effective Date Process Flow Diagram

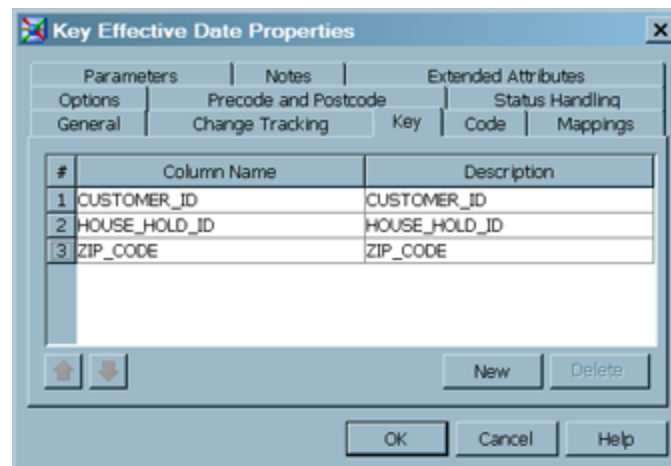


#### Identify Source Columns

Perform the following steps to identify the primary key and datetime columns in the transformation:

1. Open the properties window of the Key Effective Date transformation, and then select the **Change Tracking** tab.

2. Under **Column Name**, triple-click to open the pull-down list to select the source and target columns that contain the begin and end datetime values.
3. Under **Expression**, enter the expression or value that is applied when begin and end datetime values are missing from a source row.
4. Open the **Key** tab and click **New**. Under **Column**, select the name of the first column in the primary key of the source table. Similarly, select in order any other columns in the primary key. The following display depicts the completed **Key** tab for the sample job.

**Display 19.16** Order of Primary Key Columns on the Key Tab

5. Click **OK** to close the properties window.

### Run the Job and View the Output

Perform the following steps to run the job and view the output:

1. Right-click in the job and select **Run**. SAS Data Integration Studio generates code for the job and submits the code to the SAS Application Server for execution.
2. If error messages are displayed on the **Status** tab, read and respond to the messages as needed.
3. After the completion of the job, right-click the target and select **Open**. The following display shows the target table data for the sample job.

**Display 19.17** Tracked Datetime Values in the Sample Target Table

View Data: CUST_TRACKED (150 rows)				
#	CUSTOMER_ID	VALID_FROM_DTTM	VALID_TO_DTTM	REVISION_NUM
1	7369	1389107215.9	1389107216.9	1
2	7369	1389107217.9	127458489600	2
3	7499	1389107215.9	1389107218.9	3
4	7499	1389107219.9	127458489600	4
5	7505	1389107215.9	1389107220.9	5
6	7505	1389107221.9	127458489600	6

---

## Closing Out Rows in Datetime Change Tracking

### ***Problem***

In a dimension table that uses datetime change tracking, you need to close out a current row without adding a new current row for that member.

### ***Solution***

To close out a current row without changing the tracked data values in that row (and therefore adding a new current row), simply load that row without data changes and with an end datetime value that is less than the current end datetime value. The row receives the new end datetime value, which closes-out the row, without creating a new current row for that member.



## Chapter 20

# Working with Change Data Capture

---

<b>About the Change Data Capture Transformations</b> . . . . .	<b>369</b>
Change Data Capture Defined . . . . .	369
Prerequisites for Change Data Capture . . . . .	370
<b>About CDC Changed Data Tables</b> . . . . .	<b>371</b>
<b>About CDC Control Tables</b> . . . . .	<b>372</b>
<b>Capture Changed Data from Oracle</b> . . . . .	<b>372</b>
Problem . . . . .	372
Solution . . . . .	373
Tasks . . . . .	373

---

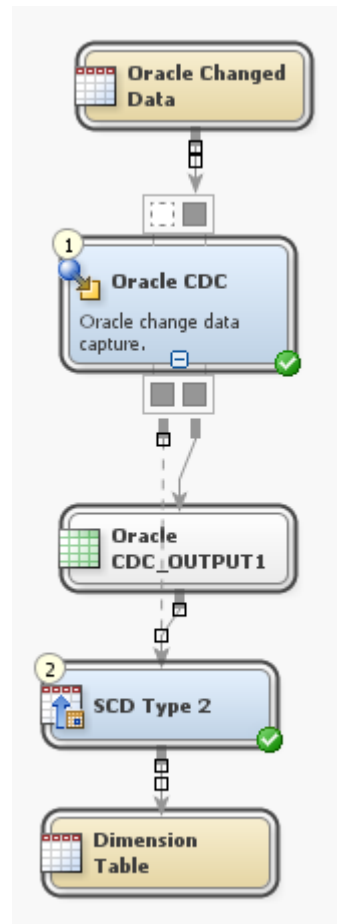
## About the Change Data Capture Transformations

### *Change Data Capture Defined*

Change data capture (CDC) is a process that shortens the time required to load data from a relational database. The process is efficient because the source is a changed data table, rather than the entire base table.

The CDC transformations in SAS Data Integration Studio are used to load dimension tables in star schemas, as part of an implementation of slowly changing dimensions. For more information, see [“About Slowly Changing Dimensions” on page 348](#).

The following diagram illustrates a job that loads changed data into a dimension table. The temporary target table that is generated by the CDC transformation is the source for the SCD Type 2 Loader transformation.

**Figure 20.1** A CDC Job that Implements Slowly Changing Dimensions

SAS Data Integration Studio provides four CDC transformations: Oracle CDC, DB2 CDC, Attunity CDC, and General CDC. The Oracle, DB2, and Attunity transformations work directly with changed data tables that are in native database format. The General CDC transformation loads change data from other vendors or from your own custom applications.

The CDC transformations are available in the Transformations tree under the heading Change Data Capture.

The CDC transformations require you to install and configure change data capture software on your relational database, and then use that software to generate changed data tables. For details, see the topic that describes the prerequisites for each of the CDC transformations.

All of the CDC transformations require you to supply a source changed data table. Additionally, the CDC transformations can be configured to read a control table. The primary purpose of the control table is to allow only one write to each record in the target. For information about control tables, see [About CDC Control Tables on page 372](#).

### Prerequisites for Change Data Capture

The CDC transformations require the following software:

#### Attunity CDC

Attunity is a data integration product, in which the Attunity Stream software enables connectivity between databases and across operating environments. The Attunity CDC

transformation has been validated on Attunity AIS 5.0 with Attunity Stream. To use the Attunity software you need to license SAS/ACCESS Interface to ODBC.

#### Oracle CDC

The Oracle CDC transformation has been validated on Oracle 10G with asynchronous CDC. The transformation requires that you license SAS/ACCESS to Oracle.

#### DB2 CDC

The DB2 CDC transformation has been validated on DB2/UDB, release 8.1, fixpak 3. The transformation requires that you license SAS/ACCESS to DB2.

#### General CDC

The General CDC transformation has no prerequisites.

---

## About CDC Changed Data Tables

In jobs that include changed data capture transformations, the source is a table that records changes to a database. Each row in the source changed data table records an insert, update, or delete action. Each row includes the data that was involved in the action.

The CDC transformations generate target data that is suitable for loading into star schemas using the SCD Type 2 Loader transformation.

The source changed data tables are generally created in native database format, using technologies that are provided by the database.

The CDC transformations require certain columns in the source changed data tables. The names and order of the following columns can vary. To identify the columns to the CDC transformations, you specify option values in properties window.

The CDC transformations require the following column definitions in the source changed data tables:

#### Application Name

identifies the application that compares the source change data records to the records in the target to test for previous updates. A typical value for this column is **SAS Data Integration Studio**. The column type is character and the length is 64.

#### Table Name

identifies the source changed data table. The column type is character and the length is 64.

#### Context

provides the unique identifiers in the target that are not to be overwritten. The column type is numeric and the length is 8. For the Oracle CDC transformation, the length is 32.

#### Rows Processed

records the number of source changed data records that were processed the last time that the job was run. The type of this column is numeric and the length is 8.

#### Timestamp

identifies the time and date when the job was run. The type of this column is numeric and the length is 8.

---

## About CDC Control Tables

In jobs that include a change data capture transformation, you can use a control table to prevent the update of target rows that were processed in an earlier run. When you run a job that uses a control table, the CDC transformation first finds in the source the most recent insert, update, or delete action for a given unique identifier (business key). The most recent source row is then compared to the prior actions that appear in the control table. If the unique identifiers match, and if the rest of the rows are identical, then the source row is a duplicate and it is not added to the target.

Control tables are optional, so you need to use one only if the source changed data table contains information that was already loaded into the target.

The control table can be in SAS format or in native database format.

Column definitions in the control table are similar to those that are required in the source changed data tables.

You can use the New Table Wizard to create control tables.

In control tables, the names and order of the following columns can vary, because you identify those columns in the properties window of the CDC transformation:

### Application Name

identifies the application that compares the source change data records to the records in the target to test for previous updates. A typical value for this column is **SAS Data Integration Studio**. The column type is character and the length is 64.

### Table Name

identifies the source changed data table. The column type is character and the length is 64.

### Context

provides the unique identifiers in the target that are not to be overwritten. The context is a character value with length of 32 for DB2, Attunity, and General. Oracle context is numeric with a length of 8.

### Rows Processed

records the number of source changed data records that were processed the last time that the job was run. This value is updated at the end of the job run, as an output from the CDC transformation. The type of this column is numeric and the length is 8.

### Timestamp

identifies the time and date when the job was run, in DATETIME16.6 format. The type of this column is numeric and the length is 8.

---

## Capture Changed Data from Oracle

### Problem

You need to load changed data from an Oracle database, with the eventual purpose of updating a dimension table in a star schema.



## Solution

Create and run a job that contains an Oracle CDC transformation. The source table contains changed data from an Oracle database. A control table is used to prevent the updates of target rows that were updated in a previous run.

The steps in the following Tasks section assume that the Oracle base table was previously loaded into the dimension table in a separate job. The example job in the task section also assumes that a third job loads the CDC target table into the dimension table using the SCD Type 2 Loader. The SCD Type 2 Loader was not included in this example job as a matter of simplicity. To see an example that uses the SCD Type 2 Loader, refer to [“Loading a Dimension Table with Type 1 and 2 Updates” on page 353](#).

The source changed data table from Oracle contains all of the inserts, updates, and deletes that have occurred since the last time the dimension table was loaded.

To accommodate database deletes, the Oracle CDC transformation calculates new end dates for the corresponding rows in the dimension table. (The dimension table retains a history of data changes by closing-out records, rather than deleting them.)

The sample job includes the following tasks:

- [“Prerequisites” on page 373](#)
- [“Create and Populate the Job” on page 373](#)
- [“Configure Row Processing” on page 374](#)
- [“Configure the Use of the Control Table” on page 375](#)
- [“Run the Job, Update the Metadata, and View the Output” on page 376](#)

## Tasks

### Prerequisites

Perform the following steps to prepare your Oracle source changed data table and control table:

1. Fulfill the prerequisites for changed data capture, as defined in [“Prerequisites for Change Data Capture” on page 370](#).
2. Use Oracle tools to create the source changed data table. Typical implementations use database triggers or log mining. Typical tools are the Oracle Data Integrator or the Oracle Log Miner.
3. Specify a library for the Oracle source table. For more information, see the *SAS Intelligence Platform: Data Administration Guide*.
4. To create the control table, select **New** ⇒ **Table**.
5. In the New Table Wizard, create a new table without columns. Specify a table name and a library, and then click **Next** until you can select **Finish**. The Oracle CDC transformation provides column definitions when you run the job.

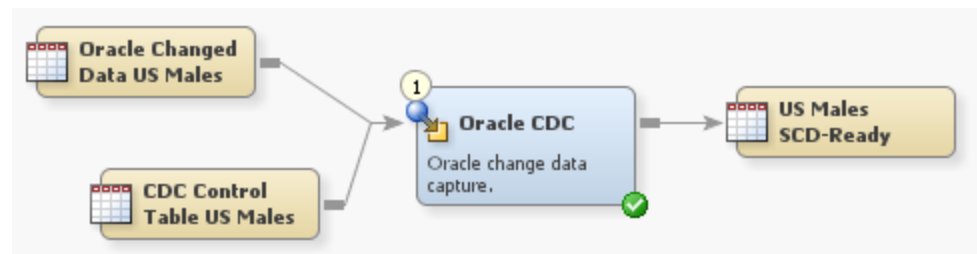
### Create and Populate the Job

Perform the following steps to create and populate a job that loads data by using an Oracle changed data table and control table:

1. Create an empty SAS Data Integration Studio job.

2. In the Transformations tree, in the **Change Data Capture** folder, drag the Oracle CDC transformation into the empty job in the **Diagram** tab.
3. Select and drag the source changed data table from its folder and drop it before the Oracle CDC transformation on the **Diagram** tab. In the example job, the source table is named Oracle Changed Data US Males.
4. Drag the cursor from the source table to the input port of the Oracle CDC transformation. This action connects the source to the transformation.
5. Select and drag the control table from its folder and drop it before the Oracle CDC transformation in the **Diagram** tab. In this example job, the control table is named CDC Control Table US Males.
6. Drag the cursor from the control table to the input port of the Oracle CDC transformation. This action connects the control table to the transformation. Note that the CDC transformation reads the control table without loading any of its data into the target.
7. Because you want to store the output of the transformation in a permanent target table, right-click the temporary work table that is attached to the transformation and select **Replace**. Then, use the Table Selector window to select the target table for the job. The target table must be registered in SAS Data Integration Studio. (For more information about temporary work tables, see [“Working with Default Temporary Output Tables” on page 126.](#))
8. Drag the target table from its folder and drop it after the Oracle CDC transformation on the **Diagram** tab. In this example, the name of the target is US Males SCD-Ready.
9. Drag the cursor from an output port of the Oracle CDC transformation to the target table. This action connects the transformation to the target. The following example shows the sample process flow.

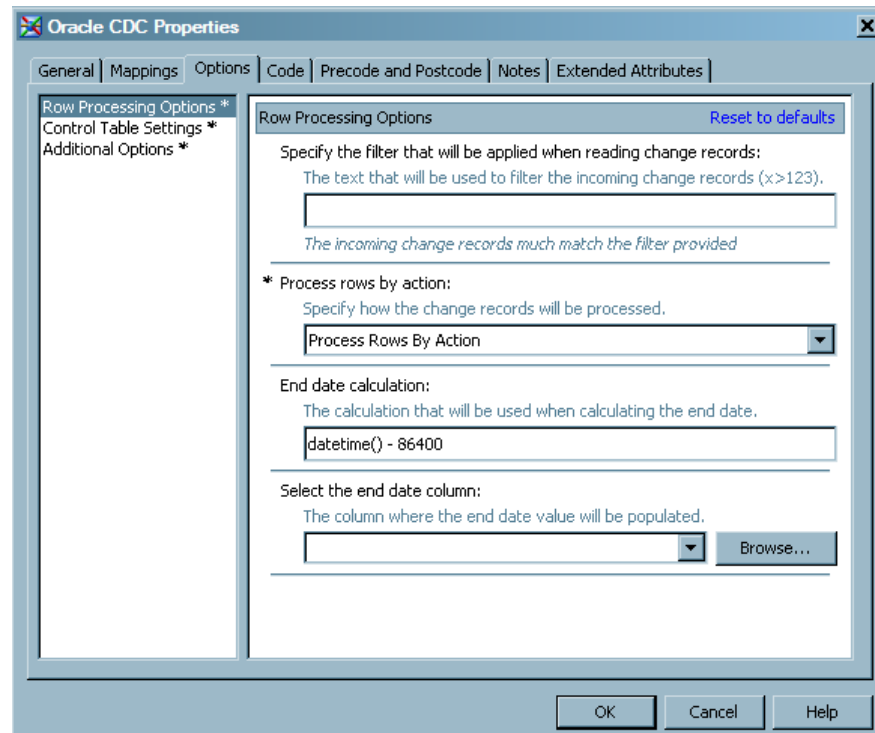
**Display 20.1** Sample Oracle CDC Process Flow Diagram



### Configure Row Processing

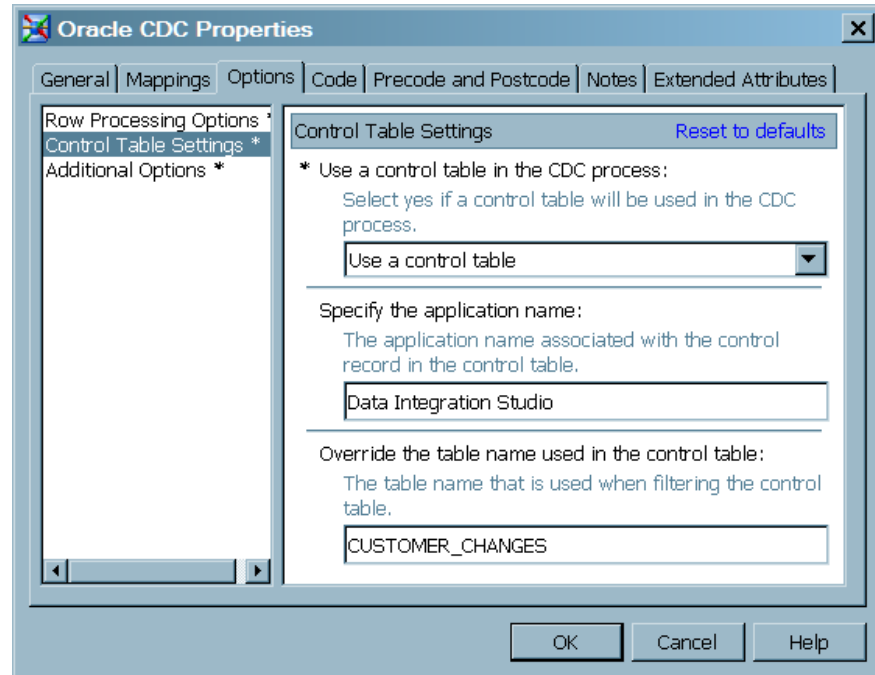
Perform the following steps to specify how rows from the source changed data table are processed for application to the target.

1. Open the properties window of the Oracle CDC transformation and select the **Options** tab.
2. For the option **Process Rows by Action**, select the value **Process Rows by Action**. Selecting this option indicates that delete processing instructions in the source changed data table are to be processed by updating an end date/time column in the target.
3. For the option **End Date Calculation**, accept the default value, which is used to calculate the date/time value that is added to the target to close-out deleted rows.
4. For the option **Select the End Date Column**, click the **Browse** button and select the numeric column that contains end date/time values. The following display depicts the completed row processing options.

**Display 20.2** Row Processing Options**Configure the Use of the Control Table**

Perform the following steps to configure the Oracle CDC transformation to use the control table.

1. On the **Options** tab, click **Control Table Settings** in the left panel.
2. For the option **Use a control table in the CDC process**, select the value **Use a control table**.
3. For the option **Specify the option name**, accept the default value **SAS Data Integration Studio**. You would enter a different application name if that application was to supply input data rows based on the contents of the source changed data table and the control table.
4. For the option **Override the table name used in the control table**, enter the name of the table that is used to filter the control table. In this example, enter the table name **CDC Control Table US Males**. You need to enter a value for this option only to use a different table when writing to and reading from the control table. The following display depicts the completed control table settings.

**Display 20.3** Completed Control Table Options

5. Click **OK** to save the option specifications and close the properties window.

### **Run the Job, Update the Metadata, and View the Output**

Perform the following steps to run the job and view the output:

1. Right-click in the job and select **Run**. SAS Data Integration Studio generates code for the job and submits the code to the SAS Application Server for execution.
2. If error messages are displayed on the **Status** tab, read and respond to the messages as needed.
3. To store the metadata for the control table columns that were created by the Oracle CDC transformation, right-click the control table on the **Diagram** tab and select **Update Metadata**. This step and the next are necessary only when you create a control table without column definitions, and only after the first time you run the job.
4. To prevent the columns in the control table from appearing in the target, right-click the **Diagram** tab and ensure that a check mark does not appear next to **Automatically Propagate Columns**.
5. After the completion of the job, right-click the target and select **Open**. The following display shows the target table data for the sample job.

**Display 20.4** CDC Columns in the Sample Target Table

View Data: CUSTOMER\_SCD\_READY\_US\_MALES

Go to row  Go

#	be	Customer_Group	Customer_Age	XIDSEQ_	operation_	cdcEndDate	rowsProcessed
1	r...	Orion Club members ...	63	691	U		0
2	e...	Orion Club Gold mem...	66	692	U		1
3	r...	Orion Club members ...	74	693	U		2
4	er...	Orion Club members ...	65	694	U		3
5	e...	Orion Club Gold mem...	64	695	U		4
6	er...	Orion Club members ...	66	696	U		5
7	er...	Orion Club members ...	71	697	U		6
8	U...	Internet/Catalog Cust...	63	698	U		7
9	r...	Orion Club members ...	72	699	U		8
10	e...	Orion Club Gold mem...	74	700	U		9
11	r...	Orion Club members ...	61	701	U		10
12	U...	Internet/Catalog Cust...	60	702	U		11



## Chapter 21

# Working with Message Queues

---

<b>About Message Queues</b> .....	<b>379</b>
<b>Prerequisites for Message Queues</b> .....	<b>380</b>
<b>Selecting Message Queue Transformations</b> .....	<b>381</b>
Problem .....	381
Solution .....	381
Tasks .....	381
<b>Processing a WebSphere Queue</b> .....	<b>382</b>
Problem .....	382
Solution .....	382
Tasks .....	382
<b>Polling a Websphere Message Queue</b> .....	<b>384</b>
Problem .....	384
Solution .....	384
Tasks .....	385
<b>Processing a Microsoft Queue</b> .....	<b>386</b>
Problem .....	386
Solution .....	386
Tasks .....	386

---

## 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 21.1** 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.

Data Transfer Type	Description
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 that is 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. However, you can create a memory overrun if you transmit a very large table or file in a Websphere message queue. For more information, see the topic on "Very Large Tables or Files In WebSphere Message Queues Can Cause Memory Overruns" in the "Usage Notes" topic in SAS Data Integration Studio Help.

The Microsoft Queue Writer transformation does not transform missing numeric values to some other value. If missing values are encountered, then an error occurs. For more information about this error and specific recommendations for avoiding it, see the topic on "Microsoft Queue Writer Transformation Does Not Transform Missing Numeric Values" in the "Usage Notes" topic in SAS Data Integration Studio Help.

---

## Prerequisites for Message Queues

The following prerequisites are required in order 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 queues; MSMQ Server for Microsoft 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 that is 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 that is running on Machine X.
- The machine that is 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: Desktop Application Administration Guide*.

*Note:* If you want to launch a SAS program to read messages from a WebSphere message queue and process them, see [“Polling a Websphere Message Queue” on page 384](#).



## 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 21.2** 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 that are 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 that are necessary to get to the messaging system are defined in SAS Management Console.
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, then the message queue content is sent to a table or a SAS 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, then the message queue content is sent to a table or a SAS 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.

## Processing a WebSphere Queue

### Problem

You want to write rows from a source table into a WebSphere message queue. Then, you need to read the messages back from the queue and write them 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 messages from the queue and populate them into a target table. Perform the following tasks to process the queue:

- “Create the Websphere Queue Writer Job” on page 382
- “Configure and Run the Websphere Queue Writer Job” on page 383
- “Verify the Websphere Queue Writer Job” on page 383
- “Create the Websphere Queue Reader Job” on page 383
- “Configure and Run the Websphere Queue Reader Job” on page 383
- “Verify the Websphere Queue Reader Job” on page 384

Text and file transfers are also supported in message queues, but these transfers are not covered in this example.

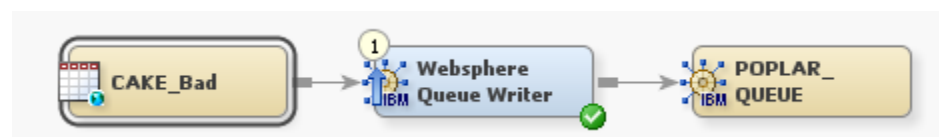
### Tasks

#### 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 Transformations tree into the empty job in the **Diagram** tab in the Job Editor window.
3. Drop the source table for the queue in the **Diagram** tab.
4. Connect the source table to the input port of the Websphere Queue Writer transformation.
5. Drop the queue from the Message queue folder in the Inventory tree in the **Diagram** tab.
6. Connect the queue to the output port of the Websphere Queue Writer transformation. The job resembles the sample shown in the following display.

**Display 21.1** 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. 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 runs 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 Writer Job

Perform the following steps to verify the results of the queue writer job:

1. Open the IBM WebSphere Queue Explorer application.
2. Select the queue that you created and ran. Then, verify that the expected messages are sitting on the queue.

### 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 Transformations tree into the empty job in the **Diagram** tab in the Job Editor window.
3. Drop the queue that you created and ran on the **Diagram** tab.
4. Connect the queue to the input port of the Websphere Queue Reader transformation.
5. Because you want to have a permanent target table to contain the output for the transformation, right-click the temporary work table that is attached to the transformation and click **Replace** in the pop-up menu. Then, use the Table Selector window to select the target table for the job. The target table must be registered in SAS Data Integration Studio. (For more information about temporary work tables, see [“Working with Default Temporary Output Tables”](#) on page 126.)
6. After these steps have been completed, the process flow diagram for this example resembles the following display.

**Display 21.2** 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 Reader transformation.

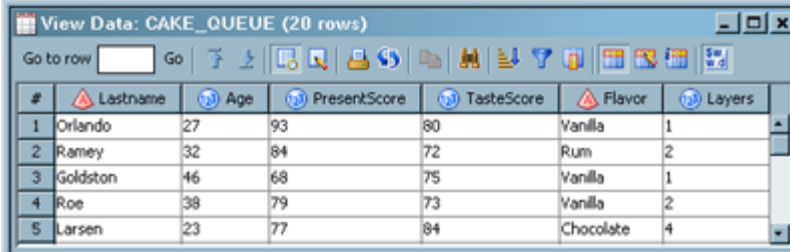
2. Select **Table** in the **Message Type** group box. Save the setting and close the properties window. Remember that you verified that the message queue contained the messages from the source table in the Verify the Websphere Queue Writer Job section.
3. Run the job. If you are prompted to do so, enter a user ID and password for the default SAS Application Server that generates and runs 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.
2. Access the View Data window for the target table. A sample target table is shown in the following example.

**Display 21.3** Sample Target Table Data



#	Lastname	Age	PresentScore	TasteScore	Flavor	Layers
1	Orlando	27	93	80	Vanilla	1
2	Ramey	32	84	72	Rum	2
3	Goldston	46	68	75	Vanilla	1
4	Roe	38	79	73	Vanilla	2
5	Larsen	23	77	84	Chocolate	4

3. 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 Reader properties window. To access this window, right-click **Websphere Queue Reader** and click **Properties** in the pop-up menu. Then, you can correct the formats as needed.

## Polling a Websphere Message Queue

### Problem

You want to launch a SAS program to read messages from a Websphere message queue and process them.

### Solution

You can create a job in SAS Data Integration Studio to read message from a queue and add appropriate transformations or SAS code to process the message. You can then deploy this job that contains a Websphere Queue Reader transformation for scheduling to be run in batch mode. Message Queue Polling Server is configured to launch this deployed job to read and process messages from the queue whenever a specified number of messages accumulates on the Websphere queue.

Once you configure a Message Queue Polling Server, you can use the object spawner to perform message queue polling to monitor queues and start SAS programs to read and process messages. The Object Spawner application can monitor the queue depth for a message queue and start a SAS program to process messages on the queue. Message queue polling enables you to configure the application monitor so that new SAS sessions can be started as needed.

Message queue polling enables load balancing across multiple SAS sessions. You can configure any number of definitions to specify which queues to monitor, the transport (MQSeries or MQSeries C), the number of messages (the queue depth) required to start a new SAS session, and the wait interval between queries. Your administrator can customize the configuration so that sufficient processes are running to handle the number of messages on the queue.

You or an administrator must perform the following tasks to create the connection between the SAS job and the Message Queue Polling Server:

1. Define the Message Queue Server and the message queue. See the "Administering Message Queues" section in the "Administering SAS Data Integration Studio" chapter of the *SAS Intelligence Platform: Desktop Application Administration Guide*.
2. Create a queue reader job. See [“Processing a WebSphere Queue” on page 382](#).
3. Deploy the queue reader job for scheduling. See [“Deploy the SAS Job for Scheduling” on page 385](#).
4. Create the Message Polling Server. Then, configure it to point to the SAS job that is used to process the message (such as the queue reader job). See the "Administering Message Queues" section in the "Administering SAS Data Integration Studio" chapter of the *SAS Intelligence Platform: Desktop Application Administration Guide*.
5. Configure the object spawner to recognize the Message Polling Server. Then, refresh the object spawner to start the polling server job. See the "Administering Message Queues" section in the "Administering SAS Data Integration Studio" chapter of the *SAS Intelligence Platform: Desktop Application Administration Guide*.

## Tasks

### **Deploy the SAS Job for Scheduling**

Perform the following steps to deploy a SAS job such as a queue reader job for scheduling and eventual linkage to a Message Polling Server:

1. Right-click the SAS job in the Folders tree. Click **Scheduling** in the pop-up menu. Then, click **Deploy** in the submenu.
2. Verify that the appropriate batch server, deployment directory, deployed job name, and location are displayed in the Deploy a job for scheduling window.
3. Click **OK** to deploy the SAS job for scheduling.

You can now use information about this deployed SAS job in your Message Polling Server configuration.

## Processing a Microsoft Queue

### Problem

You want to write rows from a file into a Microsoft message queue. Then, you need to read the messages back from the queue and write them into a target table.

### Solution

You can use the Microsoft Queue Writer transformation in SAS Data Integration Studio to write the data to the message queue. Then, you can use the Microsoft Queue Reader transformation to read the message from the queue and populate them into a target table. Perform the following tasks:

- “Create the Microsoft Queue Job” on page 386
- “Configure and Run the Microsoft Queue Job” on page 387
- “Verify the Microsoft Queue Job” on page 387

### Tasks

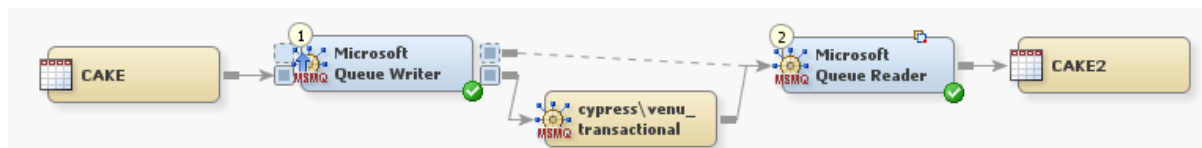
#### Create the Microsoft Queue Job

Perform the following steps to create and populate the job:

1. Create an empty job.
2. Select and drag the Microsoft Queue Writer transformation from the **Access** folder in the Transformations tree into the empty job on the **Diagram** tab in the Job Editor window.
3. Connect the source table to the input port of the Microsoft Queue Writer transformation.
4. Connect the queue to the output port of the Microsoft Queue Writer transformation.
5. Drag the Microsoft Queue Reader transformation onto the **Diagram** tab in the Job Editor window.
6. Connect the queue to the Microsoft Queue Reader transformation.
7. Because you want to have a permanent target table to contain the output for the transformation, right-click the temporary work table that is attached to the transformation and click **Replace** in the pop-up menu. Then, use the Table Selector window to select the target table for the job. The target table must be registered in SAS Data Integration Studio. (For more information about temporary work tables, see “Working with Default Temporary Output Tables” on page 126.)

The job resembles the sample shown in the following display.

**Display 21.4** Sample Microsoft Message Queue Process Flow



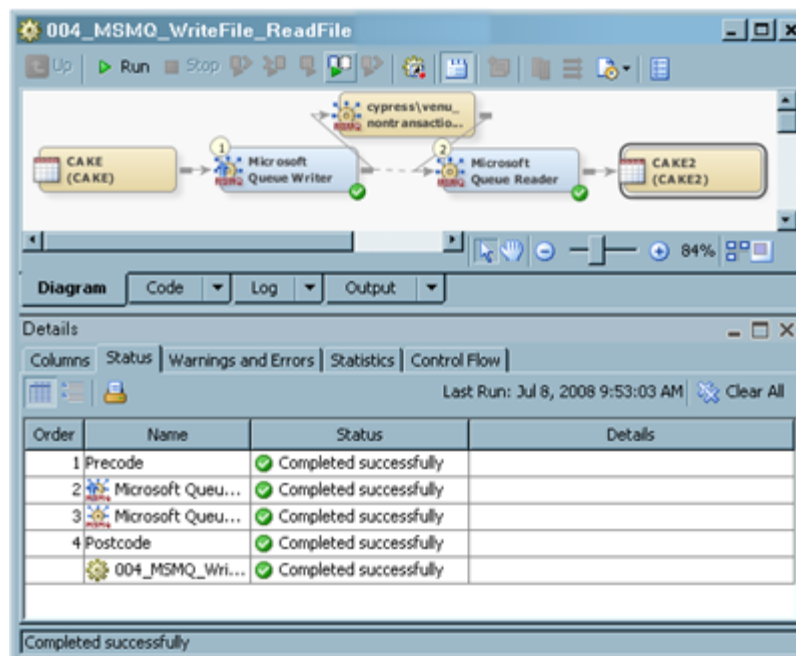
The source table for the sample job is named CAKE. The target table is named CAKE2, and the queue is named cypress\venu\nontransactional.

### Configure and Run the Microsoft Queue Job

Perform the following steps to configure the job:

1. Open the **Queue Options** tab of the properties window for the Microsoft Queue Writer transformation.
2. Specify the source for the queue. The sample job uses a file. You can also use text or a table as the source.
3. Open the **Queue Options** tab of the properties window for the Microsoft Queue Reader transformation.
4. Specify the target for the queue. The sample job uses a file. You can also use text or a table as the target.
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 runs SAS code for the job. The server executes the SAS code for the job. The following display shows that the job runs successfully.

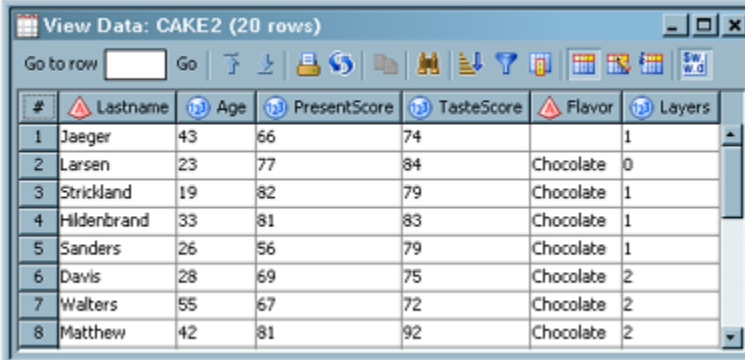
**Display 21.5** Sample Completed Microsoft Message Queue Job



### Verify the Microsoft Queue Job

Perform the following steps to verify the results of the queue job:

1. Examine the data in the source file.
2. Access the **View Data** window for the target table. A sample target table is shown in the following example.

**Display 21.6** Target Table Data for the Sample Job

#	Lastname	Age	PresentScore	TasteScore	Flavor	Layers
1	Jaeger	43	66	74	Chocolate	1
2	Larsen	23	77	84	Chocolate	0
3	Strickland	19	82	79	Chocolate	1
4	Hildenbrand	33	81	83	Chocolate	1
5	Sanders	26	56	79	Chocolate	1
6	Davis	28	69	75	Chocolate	2
7	Walters	55	67	72	Chocolate	2
8	Matthew	42	81	92	Chocolate	2

3. Confirm that the source file and the target table contain identical data. This means that the data was transferred successfully through the Microsoft message queue.



## Chapter 22

# Working with SPD Server Cluster Tables

---

<b>About SPD Server Cluster Tables</b> .....	<b>389</b>
<b>Creating an SPD Server Cluster Table</b> .....	<b>390</b>
Problem .....	390
Solution .....	390
Tasks .....	390
<b>Maintaining an SPD Server Cluster</b> .....	<b>391</b>
Problem .....	391
Solution .....	391

---

## About SPD Server Cluster Tables

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 a cluster table, 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*.
- All of the source tables that are to be added to a cluster table have been registered in the SPD Server library. All source tables must have the same column structure.
- A cluster table has been registered in the SPD Server library. The cluster table and all of its source tables must have the same column structure. One way to ensure that all of these tables have the same columns is to use the New Table wizard to copy the metadata from a source table and save it as the metadata for the cluster table. For details about using the New Table wizard, see [“Registering New Tables with the New Table Wizard” on page 66](#).

## Creating an SPD Server Cluster Table

### Problem

You want to create an SPD Server cluster table. 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 Job Editor window. For more information, see the following tasks:

- “Create and Populate the Job” on page 390
- “Specify Options for the Create or Add to a Cluster Transformation” on page 391

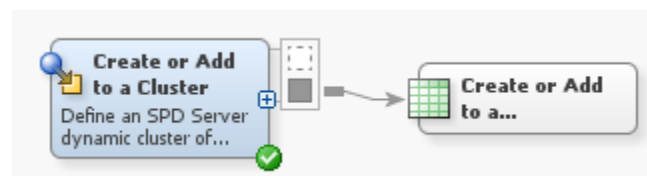
### Tasks

#### Create and Populate the Job

Perform the following steps to build a job that creates an SPD Server cluster table. If you add the List Cluster Contents transformation to the same job, you can list the source tables in the **Output** tab in the Job Editor window.

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 Job Editor window. This transformation produces a temporary output table that you can use as a permanent output table or as an input to another transformation or table loader. You can also replace the temporary output table with a permanent target table. The 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. The following example shows the temporary output table.

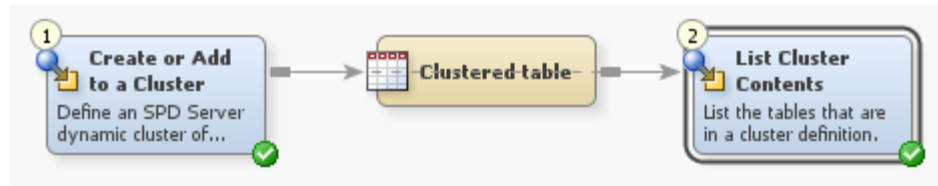
**Display 22.1** Sample SPD Server Cluster Table Job with Temporary Output Table



3. To replace the temporary output table with the clustered table, right-click the temporary work table that is attached to the Create or Add to a Cluster transformation and click **Replace** in the pop-up menu. Then, use the Table Selector window to select the cluster table. For additional information about temporary output tables, see “Working with Default Temporary Output Tables” on page 126.
4. To verify what tables were clustered, add the List Cluster Contents transformation to the process flow, and drop the transformation on the Job Editor window. Then, drag the cursor from the output port of the cluster table to the input port of the List Cluster

Contents transformation. The following display shows a process flow diagram for the resulting job. The numbers on the transformations show the order of the job's processes.

**Display 22.2** 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.

### Specify Options for the Create or Add to a Cluster Transformation

Perform the following steps to specify options for the Create or Add to a Cluster Transformation and run the job.

1. 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.
2. Limit the tables that are included in the cluster table by entering a string in the **Filter: table name contains ...** field found on the **Cluster Options** window. In this case, enter **CLUSTER** because all tables that are required include this string in the table name.
3. Enter a value into the **Set maximum number of slots** 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. If the slot size does not accommodate all of the clustered tables, then you must delete the existing cluster definition and define a new cluster that includes an adequate value for the maximum number of slots.
4. Click **OK** to save the setting and close the properties window.
5. 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 22.3** Cluster Contents on Output Tab

```
Cluster Name CLUSTEREDTABLE, Mem=CLUSTER1
Cluster Name CLUSTEREDTABLE, Mem=CLUSTER2
```

## Maintaining an SPD Server Cluster

### Problem

You want to maintain an existing SPD server cluster by adding a table to a cluster, generating a list of tables that are 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 22.1** SPD Server Transformations

Server	Tasks That Require This Server
Add a table to a cluster	<p>Perform the following steps to use the Create or Add to a Cluster transformation:</p> <ol style="list-style-type: none"> <li>1. Create an empty job.</li> <li>2. Drop the Create or Add to a Cluster transformation into the Job Editor window.</li> <li>3. Replace the temporary output table with the clustered table.</li> <li>4. Drag the cursor from the output port of the Create or Add to a Cluster transformation to the input port of the cluster table.</li> <li>5. Run the job.</li> </ol>
Generate a list of tables in a cluster	<p>Perform the following steps to use the List Cluster Contents transformation:</p> <ol style="list-style-type: none"> <li>1. Create an empty job.</li> <li>2. Drop the List Cluster Contents transformation into the Job Editor window.</li> <li>3. Drop the cluster table onto the Job Editor window.</li> <li>4. Drag the cursor from the output port of the cluster table to the input port of the List Cluster Contents transformation.</li> <li>5. Run the job.</li> </ol> <p>Note that you can also include the List Cluster Contents transformation in an SPD server cluster job. This generates a cluster list each time you create a cluster.</p>
Remove a cluster definition	<p>Perform the following steps to use the Remove Cluster transformation:</p> <ol style="list-style-type: none"> <li>1. Create an empty job.</li> <li>2. Drop the Remove Cluster transformation into the Job Editor window.</li> <li>3. Drop the cluster table into the Job Editor window.</li> <li>4. Drag the cursor from the output port of the cluster table to the input port of the Remove Cluster transformation.</li> <li>5. Run the job.</li> </ol> <p>The cluster table is now removed and the tables that were in the cluster are now available as individual tables.</p>

## Part 4

---

# Appendixes

<i>Appendix 1</i>	
<b>Main Windows and Wizards</b> .....	395
<i>Appendix 2</i>	
<b>Java Code and Methods for Report Plug-ins</b> .....	427



## Appendix 1

# Main Windows and Wizards

---

<b>Analysis Window</b> .....	<b>396</b>
<b>Checkouts Tree</b> .....	<b>397</b>
<b>Code Editor</b> .....	<b>397</b>
<b>Comparison Results Window</b> .....	<b>398</b>
<b>Connection Profile Window</b> .....	<b>399</b>
<b>Desktop</b> .....	<b>399</b>
<b>Details Pane</b> .....	<b>401</b>
<b>Expression Builder</b> .....	<b>402</b>
<b>Folders Tree</b> .....	<b>403</b>
<b>Inventory Tree</b> .....	<b>404</b>
<b>Job Editor</b> .....	<b>407</b>
<b>Properties Windows</b> .....	<b>408</b>
Basic Properties .....	408
Job Properties .....	409
Transformation Properties .....	409
Table Properties .....	410
<b>Reports Window</b> .....	<b>411</b>
<b>Tools-Options Window</b> .....	<b>412</b>
<b>Transformations Tree</b> .....	<b>413</b>
Introduction to Transformations .....	413
Overview of the Transformations Tree .....	413
Access Folder .....	414
Analysis Folder .....	414
Archived Folder .....	415
Change Data Capture Folder .....	415
Control Folder .....	416
Data Folder .....	416
Data Quality Folder .....	417
Data Transforms Folder .....	418
Output Folder .....	418
Publish Folder .....	418
SPD Server Dynamic Cluster Folder .....	419
<b>Tree View</b> .....	<b>419</b>

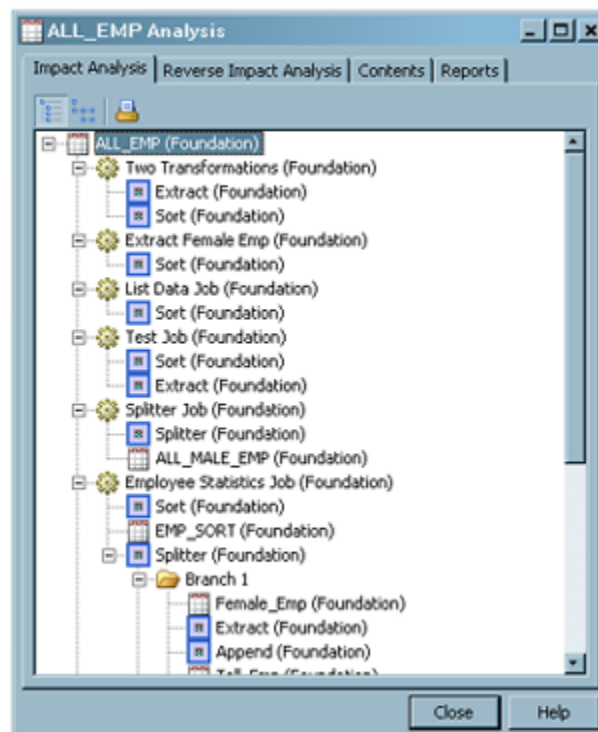
<b>View Data Windows</b> .....	<b>421</b>
View Data Window .....	421
View File Window .....	422
<b>Wizards</b> .....	<b>423</b>
New Object Wizards .....	423
Register Tables Wizards .....	424
Cube Wizards .....	424
Data Surveyor Wizards .....	425
Metadata Import and Export Wizards .....	425

---

## Analysis Window

Use the Analysis window to examine the possible impact of changing the metadata for data stores (tables, external files, and cubes), columns, generated transformations, and other objects. To access the **Analysis** window, right-click the object and select **Analyze**. The following display shows a sample **Impact Analysis** for a table.

**Display A1.1** Sample Impact Analysis for a Table



The Analysis window contains the following tabs:

- **Impact Analysis:** identifies the data stores, columns, jobs, and transformations that are *affected by* a change in a selected object. For more information, see [“Performing an Impact Analysis” on page 248](#).
- **Reverse Impact Analysis:** identifies the data stores, columns, jobs, and transformations that *contribute to* the content of a selected object. For more information, see [“Performing Reverse Impact Analysis ” on page 253](#).
- **Contents:** executes and displays the CONTENTS procedure for a selected table.



- **Reports:** enables you to run any custom analysis reports that were created for your site. If your site has not created such reports, the icons on this tab are dimmed. For more information about custom reports, see [“Example Java Code for a Report Plug-in” on page 427](#).

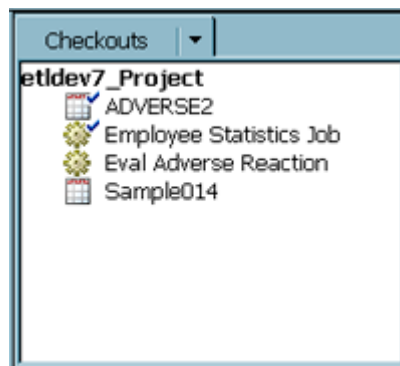
---

## Checkouts Tree

The Checkouts tree is one of the tree views in the left panel of the desktop. This tree is displayed automatically when you are working under change management in SAS Data Integration Studio. Under change management, most users are restricted from adding or updating the metadata in a change-managed folder in the Folders tree. Authorized users, however, can add new metadata objects and check them in to the change-managed folder. They can also check out metadata objects from the change-managed folder in order to update them. The objects are locked so that no one else can update them as long as the objects are checked out. When the users are ready, they check the objects in to the change-managed folder, and the lock is released.

If you are authorized to work in a change-managed folder, a Checkouts tree is added to your desktop in SAS Data Integration Studio. The following display shows a sample Checkouts tree.

**Display A1.2** Sample Checkouts Tree

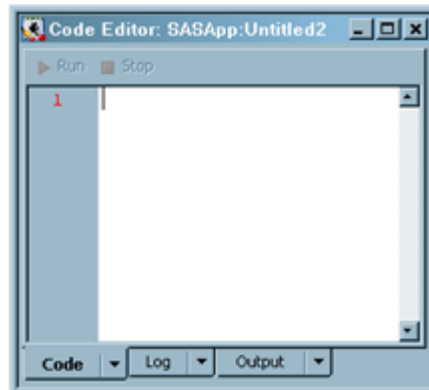


Metadata that has been checked out for update has a check mark beside it, such as the first two objects in the preceding display. New metadata objects that have never been checked in do not have a check mark beside them, such as the last two objects in the preceding display. For more information about change management, see [“Working with Change Management” on page 36](#).

---

## Code Editor

The Code Editor is a window that you can use to develop and execute SAS code. For example, you can use the Code Editor window to develop and verify user-written code, and then you can use that code to replace the generated code for a job or a transformation. The following display shows the Code Editor window.

**Display A1.3** Code Editor Window

Note that the window contains **Code**, **Log**, and **Output** tabs.

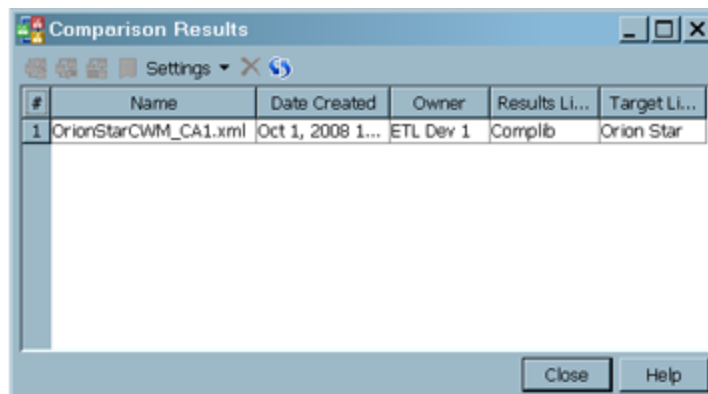
To display the Code Editor window, select **Tools** ⇒ **Code Editor** from the desktop. To submit code for execution, click **Run** on the **Code Editor** toolbar. Alternatively, you can select **Actions** ⇒ **Run** from the desktop. To display Help for the **Code Editor**, press the F1 key. To customize the appearance and behavior of the **Code Editor**, select **Tools** ⇒ **Options** from the desktop and click the **Code Editor** tab.

Any options that you specify for the Code Editor window affect the **Code** tab in the Job Editor as well.

---

## Comparison Results Window

The Comparison Results window enables you to select the results of a comparison between existing metadata and metadata that is imported with the Import Metadata Wizard. Each successful comparison operation generates a record of the result, such as the record in the next display.

**Display A1.4** Comparison Results Window

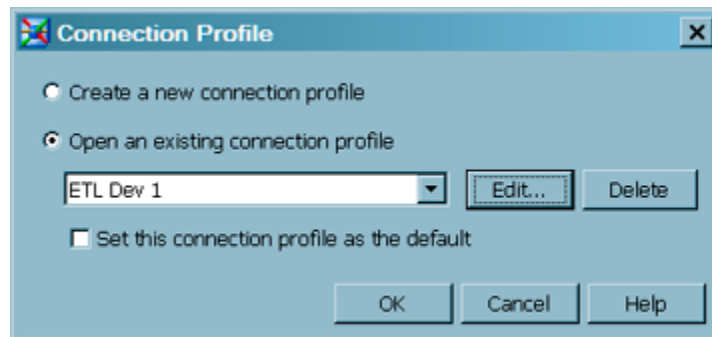
When you select a comparison result, the icons on the toolbar are activated. You can click these icons to view the differences between the imported metadata and existing metadata, or to perform other tasks. For more information, see [“Importing Updated Metadata with a SAS Metadata Bridge”](#) on page 55.

---

## Connection Profile Window

When you start SAS Data Integration Studio, the Connection Profile window displays in front of the desktop, as shown in the next display.

**Display A1.5** Connection Profile Window



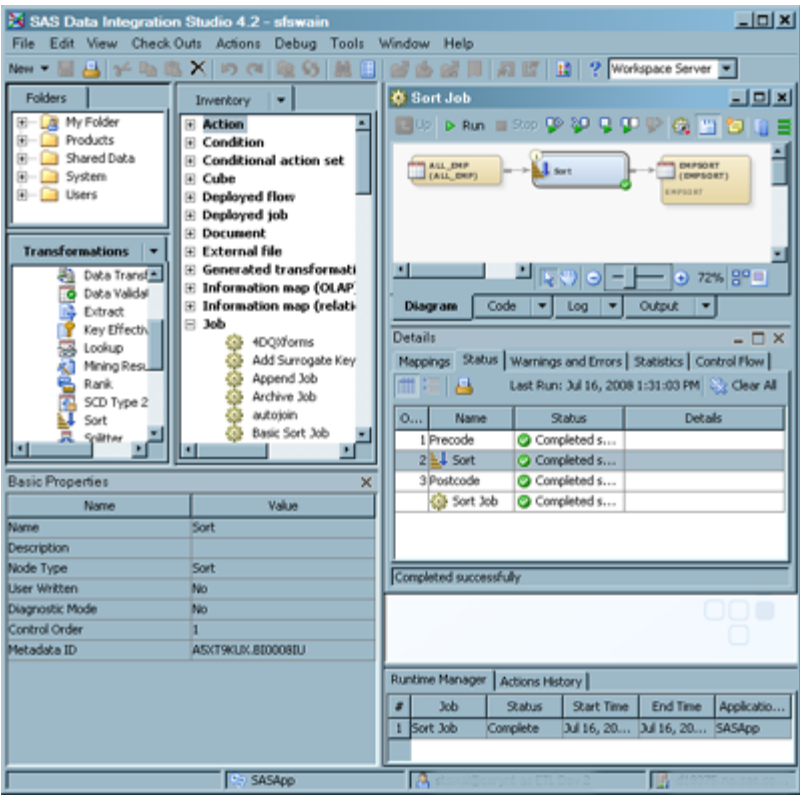
A connection profile enables you to connect to a SAS Metadata Server. You cannot do any work until you open an existing profile or create a new one. For more information, see [“Connecting to a SAS Metadata Server” on page 26](#).

---

## Desktop

After you open a connection profile, the SAS Data Integration Studio desktop displays. The following display shows a typical desktop.

**Display A1.6**    SAS Data Integration Studio Desktop



The main components of the desktop are described in the following table.

**Table A1.1**    Desktop Components

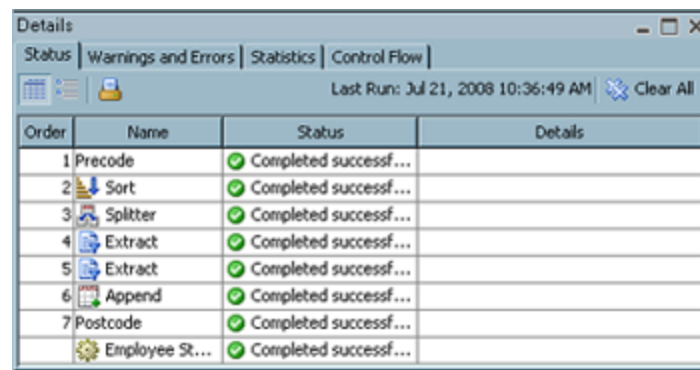
Component	Location	Description
Title bar	Top of the desktop	Shows the current version of SAS Data Integration Studio and the name of the current connection profile.
Menu bar	Under the title bar	Provides access to the drop-down menus. The list of active options varies according to the current work area and the kind of object that you select. Inactive options are disabled or hidden.
Toolbar	Under the menu bar	Provides access to 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.
Tree view	Left pane on the desktop	Provides access to the Basic Properties pane, Checkouts tree, Folders tree, Inventory tree, and Transformations tree. For more information, see “Tree View” on page 419.

Component	Location	Description
Basic Properties pane	Bottom of the left pane on the desktop	Displays basic properties of an object that is selected in the tree view. To display this pane, select <b>View</b> ⇒ <b>Basic Properties</b> from the desktop. For more information, see <a href="#">“Properties Windows” on page 408</a> .
Status bar	Bottom of the desktop	<p>Displays the name of the currently selected object, the name of the default SAS Application Server if one has been selected, the login ID and metadata identity of the current user, and the name of the current SAS Metadata Server.</p> <p>To select a different SAS Application Server, double-click the name of that server to display a dialog box.</p> <p>If the name of the SAS Metadata Server turns red, the connection is broken. In that case, you can double-click the name of the metadata server to display a dialog box that enables you to reconnect.</p>
Job Editor	Right pane of the desktop	Used to create and maintain jobs in SAS Data Integration Studio. To display this window, right-click a job in the tree view, and select <b>Open</b> . For more information, see <a href="#">“Job Editor” on page 407</a> .
Details pane	Under the Job Editor	Used to monitor and debug a job in the Job Editor. To display this pane, select <b>View</b> ⇒ <b>Details</b> from the desktop. For more information, see <a href="#">“Details Pane” on page 401</a> .
Runtime Manager	Under the Details pane	Displays the run-time status of the current job, the last time that the job was executed in the current session, and the SAS Application Server that was used to execute the job. To display this pane, select <b>View</b> ⇒ <b>Runtime Manager</b> from the desktop.
Actions History	Under the Details pane	Displays low-priority errors and warnings. To display this pane, select <b>View</b> ⇒ <b>Actions History</b> from the desktop.

---

## Details Pane

The Details pane enables you to monitor and debug a job in the Job Editor window. To display this pane, click **Details** in the Job Editor window toolbar or select **View** ⇒ **Details** from the desktop. The following display shows the **Status** tab in a typical Details pane.

**Display A1.7** Sample Details Pane

The tabs on this pane are described in the following table.

**Table A1.2** Details Pane Tabs

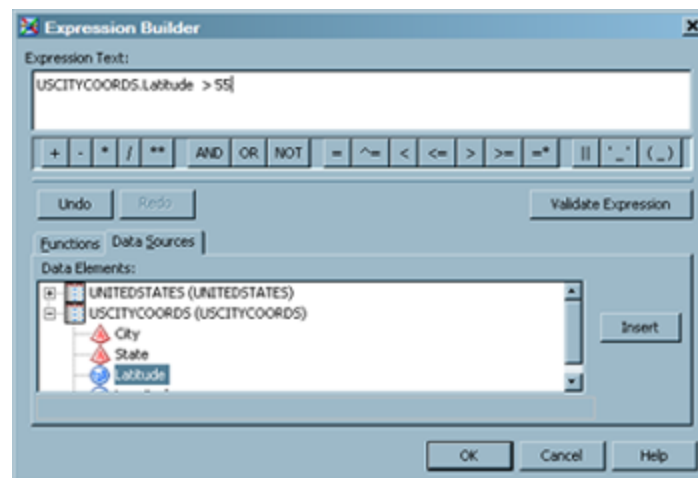
Tab	Description
<b>Status</b>	Used to display the status of each step in a submitted job.
<b>Warnings and Errors</b>	Used to display any warnings and errors that are generated when a job is submitted.
<b>Statistics</b>	Used to display run-time and table statistics that are generated by a submitted job. Includes tabular and graphical displays.
<b>Control Flow</b>	Used to display the control flow sequence of steps in a job. Also enables you to validate the control flow and change the sequence of steps.
<b>Columns</b>	Used to review and update columns in a table or external file in a job.
<b>Mappings</b>	Used to review and update mappings for transformations in a job.

The **Status**, **Warnings and Errors**, **Statistics**, and **Control** tabs are displayed whenever the Details pane is enabled for an opened job. The **Columns** tab is displayed when a table or external file in a job is selected. The **Mappings** tab is displayed when a transformation is selected.

---

## Expression Builder

The Expression Builder is a component that enables you 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 used in a WHERE clause in an SQL query.

**Display A1.8** Expression Builder Window

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:

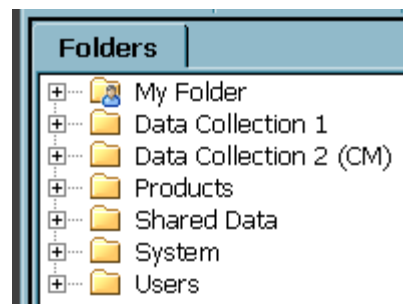
- column names
- SAS functions
- constants (fixed values)
- sequences of operands (something to be operated on like a column name or a constant) and operators, which form a set of instructions to produce a value

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*.

---

## Folders Tree

The Folders tree is one of the tree views in the left panel of the desktop. Like the Inventory tree, the Folders tree displays metadata for objects that are registered on the current metadata server, such as tables and libraries. The Inventory tree, however, organizes metadata by type and does not allow you to add custom folders. The Folders tree enables you to add custom folders.

**Display A1.9** Example Folders in the Folders Tree

For more information, see [“Working with the Folders Tree”](#) on page 27.

## Inventory Tree

The Inventory tree is one of the tree views in the left panel of the desktop. It displays metadata for objects that are registered on the current metadata server, such as tables and libraries. The Inventory tree displays a subset of the metadata that is available in the Folders tree. It displays metadata that is relevant to SAS Data Integration Studio, organized by type. For example, in the Inventory tree, you can find job metadata in the folder named **Jobs**, and so on.

*Note:* Not all metadata objects in the Inventory tree can be added or updated in SAS Data Integration Studio. Some objects appear in the tree view for other reasons.













For example, you cannot add or update actions, conditions, or deployed flows in SAS Data Integration Studio, but they appear in the tree view so that they can be included in the import and export of jobs. Likewise, you cannot add or update information maps in SAS Data Integration Studio, but they appear in the tree view so that they can be displayed in impact analysis.









The following table describes the folders and icons for metadata objects in the Inventory tree and the Folders tree.

**Table A1.3** Main Icons for Metadata Objects in the Inventory Tree and Folders Tree

Folder Name	Icon	Description
Action		Metadata for a Status Handling action. SAS provides a number of actions, such as <b>Skip the Record</b> and <b>Send Email</b> , that can be performed when certain conditions are met during the execution of a job. You cannot add or update actions.
Condition		Metadata for a Status Handling conditions. SAS provides a number of conditions, such as <b>Successful</b> and <b>Error in Process</b> , that can be tested for when jobs are executed. You cannot add or update conditions.
Conditional action set		Metadata for the default Status Handling conditional action sets (conditions and actions). You cannot add or update conditional action sets.







Folder Name	Icon	Description
Cube		Metadata for a SAS cube, a logical set of data that is organized and structured in a hierarchical, multidimensional arrangement. A cube supports online analytical processing (OLAP).
Deployed flow		Metadata for a job flow used for scheduling. Job flows are maintained in SAS Management Console. You cannot use SAS Data Integration Studio to add or update a deployed flow.
Deployed job		Metadata for a file that contains the code of a job that was deployed for scheduling. The icon for the original job has a blue triangle overlay, which indicates that the job has been deployed for scheduling.
Document		Metadata for a document. Many metadata objects have a Description attribute, which is limited to 200 characters. A document can be used to supplement the Description. Documents can contain graphics as well as text.
External file		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.
Generated transformation		Metadata for a transformation that is created with the Transformation Generator wizard. The wizard helps you specify SAS code for the transformation.
Information map (OLAP)		Metadata for an Information Map that is based on a SAS cube. Information Maps are created and maintained in SAS Information Map Studio, and they can be used in end-user applications. You cannot use SAS Data Integration Studio to add or update an information map, but information maps are shown in impact analysis.
Information map (Relational)		Metadata for an Information Map that is based on one or more tables.
Job		Metadata for a SAS Data Integration Studio job. A job is collection of SAS tasks that create output.
Job (cube)		Metadata for a read-only job that creates a SAS cube.
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.
Message queue		Metadata for a message queue. A message queue is a place where one program can send messages to be retrieved by another program.

Folder Name	Icon	Description
Mining Results		Metadata for the output of a Mining Results transformation.
Note		Metadata for a note. Many metadata objects have a Description attribute, which is limited to 200 characters. A note can be used to supplement the Description. Notes can contain text only.
OLAP Schema		Metadata for an OLAP schema. In general, do not add or update OLAP Schemas in SAS Data Integration Studio.
Prompt		Metadata for prompts. In general, do not add or update prompts in SAS Data Integration Studio.
Prompt group		Metadata for prompt groups. In general, do not add or update prompt groups in SAS Data Integration Studio.
Stored Processes		Metadata for a stored process that was generated from a SAS Data Integration Studio job. Enables users to execute SAS Data Integration Studio jobs from applications such as SAS Enterprise Guide or a Web Service client.
Table		Metadata for a table.
Web service (generated)		Metadata for generated Web services.

A modifier icon called an icon overlay indicates that an object is in a certain state or has special attributes. The following table describes the overlay icons for metadata objects in the Inventory tree and the Folders tree.

**Table A1.4** Icon Overlays for Metadata Objects in the Inventory Tree and Folders Tree

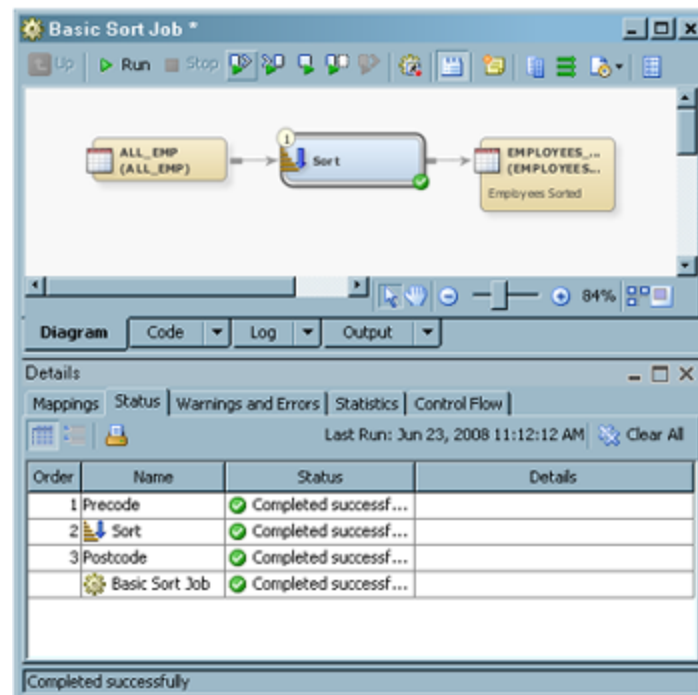
Icon Overlay	Description
	An ampersand on the icon for a table, external file, or job indicates that some attributes of the object, such as its physical path, are specified as variables rather than literal values. These parameterized tables and jobs are often used in iterative jobs.
	A blue triangle overlay on the icon for a job indicates that this job has been deployed for scheduling.
	A blue sphere overlay on the icon for a table indicates that this table has been configured as a Web stream and is the input or output of a Web service job.

Icon Overlay	Description
	A check mark overlay on any metadata object means that the object has been checked out under change management. Only the person who checked out the object can modify it.

## Job Editor

The Job Editor window enables you to create, maintain, and troubleshoot SAS Data Integration Studio jobs. To display this window, right-click a job in the tree view and select **Open**. The following display shows a sample Job Editor window.

**Display A1.10** Sample Job Editor Window



The following table describes the main tabs in the Job Editor window.

**Table A1.5** Job Editor Tabs

Tab	How to Display the Tab	Description
<b>Diagram</b>	Always displayed.	Used to build and update the process flow for a job.
<b>Code</b>	Select <b>Tools</b> ⇌ <b>Options</b> from the desktop. On the <b>General</b> tab, select <b>Show Code Tab</b> .	Used to review or update code for a job.

Tab	How to Display the Tab	Description
<b>Log</b>	Select <b>Tools</b> ⇒ <b>Options</b> from the desktop. On the <b>General</b> tab, select <b>Show Log Tab</b> .	Used to review the log for a submitted job.
<b>Output</b>	Select <b>Tools</b> ⇒ <b>Options</b> from the desktop. On the <b>General</b> tab, select <b>Show Output Tab</b> .	Used to review the output of a submitted job.

The following table describes a number of panes that can be used with the Job Editor window.

**Table A1.6** *Panes Used with the Job Editor*

Pane	How to Display	Description
Details	Select <b>View</b> ⇒ <b>Details</b> from the desktop.	Used to monitor and debug a job in the Job Editor. For more information, see <a href="#">“Details Pane” on page 401</a> .
Runtime Manager	Select <b>View</b> ⇒ <b>Runtime Manager</b> from the desktop.	Displays the run-time status of the current job, the last time that the job was executed in the current session, and the SAS Application Server that was used to execute the job. This information is available as long as the job is active.
Actions History	Select <b>View</b> ⇒ <b>Actions History</b> from the desktop.	Displays low-priority errors and warnings.

## Properties Windows

### **Basic Properties**

The Basic Properties pane is an optional pane that can be displayed on the right side of the desktop. It displays the main attributes of an object that is selected in a tree view. To display or hide this pane, select or deselect **View** ⇒ **Basic Properties** from the menu bar.

*Note:* If you have not selected a default SAS Application Server, and you select a table in a tree view, you are prompted to select a SAS Application Server so that the Basic Properties pane can display a row count for the table. To avoid this prompt, you can select a default SAS Application Server for SAS Data Integration Studio, or you can hide the Basic Properties pane. You can also select **Tools** ⇒ **Options** from the desktop menu bar and deselect the row count option on the **General** tab.

## Job Properties

The job properties window enables you to view or update the metadata for a SAS Data Integration Studio job. One way to display this window is to right-click a job in the Folders tree or Inventory tree, and click **Properties** in the pop-up menu. The next table describes the purpose of each tab in a job properties window. For more information about each tab, see the Help for that tab.

**Table A1.7** Tabs in a Job Properties Window

Tab	Description
<b>General</b>	Enables you to enter general information that identifies, describes, and locates the job.
<b>Code</b>	Enables you to review and modify the code that is generated for the job.
<b>Precode and Postcode</b>	Enables you review and modify user-written code that is inserted at the beginning or end of the job.
<b>Status Handling</b>	Enables you to review and modify status handling conditions and actions for the job. (Some transformations have this tab as well.)
<b>Parameters</b>	Enables you to review and modify parameters for the job.
<b>Options</b>	Enables you to review and modify options for the job.
<b>Notes</b>	Enables you to review and modify notes for the job.
<b>Extended Attributes</b>	Enables you to review and modify extended attributes for the job.
<b>Authorization</b>	Enables you to review and modify metadata access settings for the job.

## Transformation Properties

The transformation properties window enables you to view or update the metadata for a transformation in a SAS Data Integration Studio job. One way to display this window is to open a job in the Job Editor, right-click a transformation on the **Diagram** tab, and click **Properties** in the pop-up menu. The property window for most transformations has one or more tabs that are unique to that transformation. The following table describes the purpose of the common tabs for a transformation. For more information about each tab, see the Help for that tab.

**Table A1.8** Common Tabs in a Transformation Properties Window

Tab	Description
<b>General</b>	Enables you to enter general information that identifies and describes the transformation.
<b>Mappings</b>	Enables you to review and modify the mappings for the transformation.
<b>Options</b>	Enables you to review and modify options for the transformation.
<b>Code</b>	Enables you to review and modify the code that is generated for the transformation.
<b>Precode and Postcode</b>	Enables you review and modify user-written code that is inserted at the beginning or end of the transformation.
<b>Parameters</b>	Enables you to review and modify parameters for the transformation.
<b>Notes</b>	Enables you to review and modify notes for the transformation.
<b>Extended Attributes</b>	Enables you to review and modify extended attributes for the transformation.

## Table Properties

The table properties window enables you to view or update the metadata for the table. One way to display this window is to right-click a table in the Folders tree or the Job Editor and click **Properties** in the pop-up menu. The next table describes the purpose of each tab in a table properties window. For more information about each tab, see the Help for that tab.






**Table A1.9** Tabs in a Table Properties Window

Tab	Description
<b>General</b>	Enables you to enter general information that identifies and describes the table.
<b>Columns</b>	Enables you to maintain column metadata.
<b>Indexes</b>	Enables you to review, add, and modify indexes on table columns.
<b>Keys</b>	Enables you to review, add, and modify key columns.
<b>Parameters</b>	Enables you to review and modify parameters for the table.

Tab	Description
<b>Physical Storage</b>	Enables you to specify the format and location of a table.
<b>Notes</b>	Enables you to review and modify notes for the table.
<b>Extended Attributes</b>	Enables you to review and modify extended attributes for the table.
<b>Authorization</b>	Enables you to review and modify metadata access settings for the table.

The following table lists icons that represent columns and related attributes. These icons are displayed in the **Mappings** tab, **Columns** tab, the **Indexes** tab, or the **Keys** tab in the property window for tables.

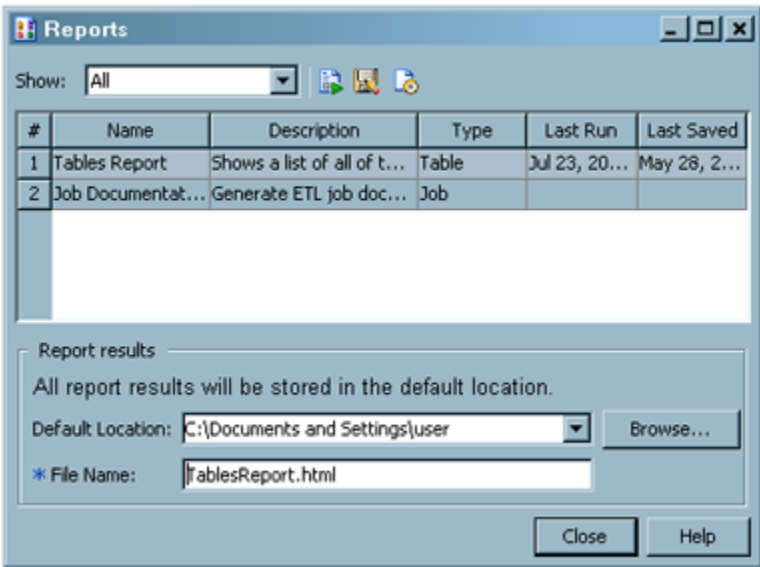
**Table A1.10** *Icons and Attributes*

Category	Icon	Description
Column		Metadata for a character column.
		Metadata for a numeric column.
Index		Metadata for an index.
Key		Metadata for a foreign key
		Metadata for a primary key or a unique key.

## Reports Window

The Reports window enables you review and run reports about your data. It also enables you to create custom reports that support your business processes and needs. To access the window, click **Reports** in the **Tools** menu or click **Reports** on the SAS Data Integration Studio toolbar. The following display shows a sample Reports window.

**Display A1.11**   Sample Reports Window



*Note:* Reports window includes report selection tools; a toolbar with controls for running, saving, and formatting reports; a table that lists available reports; and fields that enable you to specify default locations and filenames for report results.

---

## Tools-Options Window

The Options window is used to specify global options for SAS Data Integration Studio. To display this window, select **Tools** ⇨ **Options** from the desktop. The following table describes the purpose of each tab in the Options window. For more information about each tab, see the Help for that tab.

**Table A1.11**   Option Window Tabs

Tab	Description
<b>General</b>	Specifies general user interface options, such as whether SAS Data Integration Studio should prompt before discarding changes to metadata.
<b>Job Editor</b>	Specifies interface options for the Job Editor, such as the default zoom level, or whether the metadata for columns and column mappings should be automatically propagated in a process flow.
<b>Code Editor</b>	Specifies interface options for the <b>Code</b> tab in the Job Editor window, such as whether to display line numbers.
<b>SAS Server</b>	Specifies the default SAS Application Server for SAS Data Integration Studio and enables you to set options for submitting jobs to a grid.
<b>View Data</b>	Specifies interface options for the View Data window, such as whether View Data should prompt before proceeding with a lengthy navigation operation.



Tab	Description
<b>Code Generation</b>	Specifies how SAS Data Integration Studio generates code for new jobs. For example, you can specify whether optional macro variables should be added to the code that is generated for new jobs.
<b>Data Quality</b>	Specifies options that are used by the transformations in the <b>Data Quality</b> folder of the Transformations tree. For example, you can specify the location of the DQ setup file.

## Transformations Tree

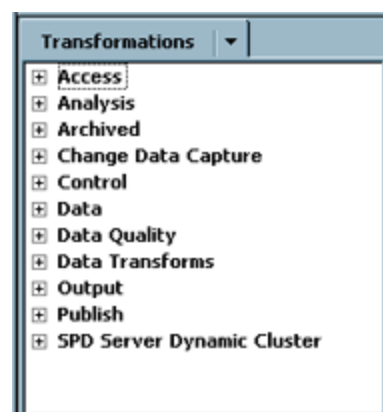
### Introduction to Transformations

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.

### Overview of the Transformations Tree

The Transformations tree organizes transformations into a set of folders. You can drag a transformation from the Transformations tree to the Job Editor, where you can connect it to source and target tables and update its default metadata. 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 display shows the standard Transformations tree.

**Display A1.12** Transformations Tree



This document has an example of the main transformations used in SAS Data Integration Studio, and the online Help has an example of all transformations. The following sections describe the contents of the Transformations tree folders.

## Access Folder

The following table describes the transformations in the **Access** folder in the Transformations tree.

**Table A1.12** Access Folder Transformations

Name	Description
File Reader	Reads an external file and writes to a SAS or DBMS table.
File Writer	Reads a SAS or DBMS table and writes to an external file.
Library Contents	Generates an output table that lists the tables contained in an input library. If there is no input library, then the transformation generates a list of tables from all of the libraries that are allocated on the SAS Workspace Server.
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 SAS 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. 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. Provides more loading options than other transformations that create tables.
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 SAS 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 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 Transformations tree.

**Table A1.13** Analysis Folder Transformations

Name	Description
Correlations	Creates an output table that contains correlation statistics.
Distribution Analysis	Creates an output table that contains a distribution analysis.
Frequency	Creates an output table that contains frequency information.
One-Way Frequency	Creates a one-way output table that contains frequency information about the relationship between two classification variables.
Summary Statistics	Creates an output table that contains summary statistics.
Summary Tables	Creates an output table that contains descriptive statistics in tabular format, using some or all of the variables in a data set. It computes many of the same statistics that are computed by other descriptive statistical procedures such as MEANS, FREQ, and REPORT.

### Archived Folder

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 transformations in the Process library. The process library continues 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 table describes the deprecated and archived transformations in the **Archived Transforms** folder in the Transformations tree.

**Table A1.14** Archived Transforms Folder Transformations

Name	Description
Fact Table Lookup	Loads source data into a fact table and translates business keys into generated keys.  This older transformation is marked with a flag on its icon. This flag indicates that the transformation is an older version of an updated transformation.

### Change Data Capture Folder

Change data capture (CDC) is a process that shortens the time required to load data from relational databases. The CDC loading process is more efficient because the source table contains changed data only. The changed data table is much smaller than the relational base table. The following table describes the transformations in the **Change Data Capture** folder in the Transformations tree.

**Table A1.15** Change Folder Transformations

Name	Description
Attunity CDC	Loads changed data only from Attunity and other selected databases.
DB2 CDC	Loads changed data only from DB2 databases.
General CDC	Loads changed data only from a wide range of databases.
Oracle CDC	Loads changed data only from Oracle databases.

## Control Folder

The following table describes the transformations in the **Control** folder in the Transformations tree.

**Table A1.16** 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 Folder

The following table describes the transformations in the **Data Transforms** folder in the Transformations tree.

**Table A1.17** Data Folder Transformations

Name	Description
Append	Creates a single target table by combining data from several source tables.
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.

Name	Description
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.
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.
Rank	Ranks one or more numeric column variables in the source and stores the ranks in the target.
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.
Sort	Reads data from a source, sorts it, and writes the sorted data to a target.
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.
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 or columns.
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 analyze metadata for other transformations.

### Data Quality Folder

The following table describes the transformations in the **Data Quality** folder in the Transformations tree.

Name	Description
------	-------------

Apply Lookup Standardization	Applies one or more schemes to one or more columns in a source table. Applying schemes modifies your source data according to rules that are defined in the schemes.
Create Match Code	Establish relationships between source rows. You can create match codes at specified levels of sensitivity. You can also assign cluster numbers to groups of source rows that generate the same match codes.
DataFlux IS Job	Executes jobs services on DataFlux Integration Servers from DataFlux, a SAS company. Used to cleanse larger amounts of source data.
DataFlux IS Service	Executes services on DataFlux Integration Servers from DataFlux, a SAS company. Used to synchronously process smaller amounts of data, in coordination with client applications that await a response from the server.

### Data Transforms Folder

The **Data Transforms** folder contains any transformations that have been created with the Transformation Generator wizard and not assigned to a transformation category. The folder is displayed only when a generated transformation is present. It is displayed only to other users when the generated transformations are placed in the **Shared Data** folder.

### Output Folder

The following table describes the transformations in the **Output** folder in the Transformations tree.

**Table A1.18** Output Folder Transformations

Name	Description
List Data	Creates an HTML report that contains selected columns from a source table.

### Publish Folder

The following table describes the transformations in the **Publish** folder in the Transformations tree.

**Table A1.19** 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 Transformations tree.

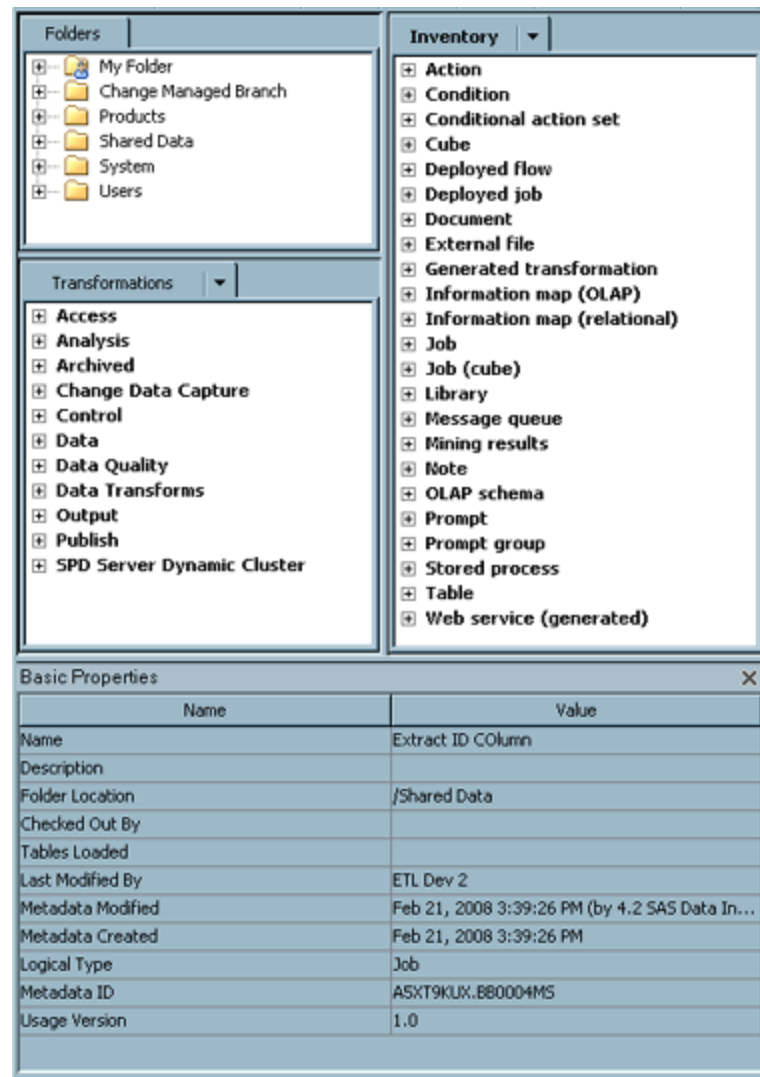
**Table A1.20** *SPD Server Dynamic Cluster Folder Transformations*

<b>Name</b>	<b>Description</b>
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.

---

## **Tree View**

The tree view is displayed on the left side of the desktop. The following display shows the tree view.



The tree view can display the following components.

**Table A1.21** Tree View Components

Component	How to Display the Component	Description
Folders tree	Displays by default.	Organizes metadata into folders that are shared across a number of SAS applications. <b>My Folder</b> and <b>Shared Data</b> are the folders that you use most of the time. For more information, see <a href="#">“Folders Tree” on page 403</a> .



Component	How to Display the Component	Description
Inventory tree	Displays by default.	Displays metadata for objects that are registered on the current metadata server, such as tables and libraries. Metadata can be accessed in folders that group metadata by type, such as Table, Library, and so on. For more information, see <a href="#">“Inventory Tree” on page 404</a> .
Transformations tree	Displays by default.	Displays transformations that can be dragged and dropped into SAS Data Integration Studio jobs. For more information, see <a href="#">“Transformations Tree” on page 413</a> .
Basic Properties pane	Select <b>View</b> ⇒ <b>Basic Properties</b> from the desktop.	Displays the basic properties of an object selected in a tree view.
Checkouts tree	Displays automatically when you are working under change management.	Displays metadata that has been checked out for update, as well as any new metadata that has not been checked in. For more information, see <a href="#">“Checkouts Tree” on page 397</a> .

---

## View Data Windows

### View Data Window

The View Data window is available in the tree views on the desktop and in process flows in the Job Editor. 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 and must exist in physical storage.

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 SAS engine library that is assigned to a SAS 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 A1.13** View Data Window

#	StockNumber	Name	Description	ListPrice	Cost	QtyOnHand	ManufacturerID
1	1	Computer ...	Four feet long - ...	129.99	99.99	5	1001
2	2	Laser Printer	Prints 30 sheets ...	299.99	299.99	5	1002
3	3	Zip Disks	Package of three	19.99	14.99	12	1002
4	4	Hanging Fo...	Package of 25	7.99	4.99	25	1003
5	5	Calculator	Scientific with gr...	25.99	18.99	7	1004

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.

## 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. It 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.

The following display shows a typical View File window.

**Display A1.14** View File Window

	NAME, GENDER, SEX, AGE, HEIGHT, WEIGHT
2	Alfred, 2, 14, 69, 11.25
3	Alice, 1, 13, 56.5, 84
4	James, 2, 12, 57.3, 83
5	Pat, 3, 11, 51.3, 50.5
6	Terry, 3, 16, 72, 150

## Wizards

### New Object Wizards

Most new object wizards enable you to register objects, such as libraries and tables, so that they can be used in SAS Data Integration Studio jobs. One way to display these wizards is to right-click an appropriate destination folder in the **Folders** tree, then select **New** ⇒ **Folder**, or **New** ⇒ **Job**, and so on. The next table describes the new object wizards.

**Table A1.22** New Object Wizards

Wizard	Description
New Folder	Adds a folder in the <b>Folders</b> tree.
New Job	Adds metadata for a new SAS Data Integration Studio job.
New Table	Registers a single table that does not yet exist in physical storage, such as a table that is created when a job is executed for the first time.
New Transformation	Adds a new generated transformation. The wizard guides you through the steps of specifying SAS code for the transformation and saving the transformation to the metadata server. After the transformation is saved, it displays in the Transformations tree, Folders tree, and Inventory tree where it is available for use in any job.
New External File	Registers an external file, which 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.
New Library	Registers a SAS library.
New Document	Registers a document that you can associate with one or more metadata objects.
New Note	Creates and registers a note that you can associate with one or more metadata objects.
New Cube (Cube Designer in add mode)	Creates and registers a SAS OLAP cube.
New OLAP Schema	Changes the OLAP schema associated with a cube.

## Register Tables Wizards

Register Tables wizards enable you to register one or more selected tables, based on the physical structure of the tables. One way to display these wizards is to right-click an appropriate destination folder in the **Folders** tree, and then select **Register Tables**. Another way is to right-click the icon for the library that contains the physical tables, and then select **Register Tables**.

## Cube Wizards

Cube wizards enable you to create and maintain SAS OLAP cubes. 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 create and maintain the cube. The main cube wizards in SAS Data Integration Studio are described in the following table.

**Table A1.23** Cube Wizards

Wizard	How to Display the Wizard	Description
Cube Designer (add mode)	Right-click an appropriate destination folder in the <b>Folders</b> tree, and then from the desktop select <b>New</b> ⇒ <b>Cube</b> .	Creates and registers a SAS OLAP cube.
Cube Designer (update mode)	Right-click a cube, and then select <b>Edit Cube Structure</b> .	Maintains a cube.
Aggregation Tuning	Right-click a cube, and then select <b>Aggregation Tuning</b> .	Updates aggregations.
View Cube	Right-click a cube, and then select <b>View Cube</b> .	Displays the cube.
Export Code	Right-click a cube, and then select <b>Export Code</b> .	Saves the code for the cube to a file.
Calculated Members	Right-click a cube, and then select <b>Maintain</b> ⇒ <b>Calculated Members</b> .	Adds, edits, and deletes the calculated members associated with the cubes that are registered to the current metadata server.
Change OLAP Schema	Right-click a cube, and then select <b>Maintain</b> ⇒ <b>Change OLAP Schema</b> .	Changes the OLAP schema associated with a cube.

For more information about SAS OLAP cubes, see *SAS OLAP Server User's Guide*.

## Data Surveyor Wizards

An optional data surveyor wizard enables you to extract, search, and navigate data from the SAP ERM system. One way to display the SAP data surveyor is to right-click an appropriate destination folder in the **Folders** tree, select **Register Tables**, and then select the data surveyor.

Optional Composite Software provides access to ERM systems such as Siebel, PeopleSoft, Oracle Applications, and Salesforce.com. For details about Composite Software and the data surveyor wizard for SAP ERM systems, see the *SAS Intelligence Platform: Data Administration Guide*.

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*.

## Metadata Import and Export Wizards

SAS Data Integration Studio enables you to import and export metadata in SAS Open Metadata Architecture format or in a format that is supported by a SAS Metadata Bridge.

The SAS Package wizards enable you to import and export metadata in SAS Open Metadata Architecture format. For example, you could use the SAS Package wizards to export a job from SAS Data Integration Studio in a test environment, and then import that job into SAS Data Integration Studio in a production environment.

**Table A1.24** SAS Package Wizards

Wizard	How to Display the Wizard	Description
Export SAS Package	In the Folders tree, right-click one or more objects to be exported, and then select <b>Export</b> ⇒ <b>SAS Package</b> .	Exports selected metadata objects to a SAS Package (SPK) file.
Import SAS Package	In the Folders tree, right-click a destination folder, and then select <b>Import</b> ⇒ <b>SAS Package</b> .	Imports SAS metadata that was exported to a SAS Package (SPK) file.

The Import and Export Metadata wizards enable you to work with metadata in a format that is supported by a SAS Metadata Bridge. You must license the appropriate bridge.

**Table A1.25** Import and Export Metadata Wizards

Wizard	How to Display the Wizard	Description
Export Metadata	In the Folders tree, right-click one or more objects to be exported, and then select <b>Export</b> ⇒ <b>Metadata</b> .	Exports table metadata that you select. You can export metadata in a format that is supported by a SAS Metadata Bridge.

Wizard	How to Display the Wizard	Description
Import Metadata	In the Folders tree, right-click a destination folder, and then select <b>Import</b> ⇨ <b>Metadata</b> .	Imports metadata 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.

## Appendix 2

# Java Code and Methods for Report Plug-ins

---

Example Java Code for a Report Plug-in .....	427
Reporting Interface Methods .....	433

---

## Example Java Code for a Report Plug-in

The simplest way to create a new report is to extend the abstract class, `AbstractReport`. `AbstractReport` is located in the `sas.dbuilder.util.jar` at `com.sas.wadmin.reports`. `AbstractReport` provides the default implementation of the majority of the methods required by the report plug-in interface, `ReportingInterface`. In this example, all of the logic to generate the report is handled by SAS code, which is embedded in the Java code. The SAS Code is submitted to the application server by the reporting framework when the **Run and view a report** button is pressed in the Reports window.

Specific JAR files are needed to use the import statements in the sample code that is provided in this section. The JAR files for your reports plug-in code are located in the folder named `SASVersionedJarRepository`. This folder is usually located in the same directory as the `SASDataIntegrationStudio` folder. Make sure your path includes the following JAR files:

```
sas.dbuilder.util.jar
sas.framework.workspace.jar
sas.oma.joma.rmt.jar
```

The following example creates a summary report of all of the tables in the metadata server. This example generates the Tables Report, which you can find in the table in the Reports window.

### CAUTION:

**The following code sample was formatted so that each line fits within the page margins even if that string was continued on another line. Newline characters within a string generate compile errors, so do not put strings on a separate line as shown in this example.**

```
import com.sas.metadata.remote.MdException;
import com.sas.wadmin.plugins.ReportingInterface;
import com.sas.wadmin.reports.AbstractReport;
import com.sas.wadmin.reports.ReportingController;
import com.sas.workspace.MessageUtil;
import com.sas.workspace.SASCodeGeneration;
import com.sas.workspace.WAdminResource;
import com.sas.workspace.WsAppServer;
```

```

import com.sas.workspace.WsServerRequest;

/**
 * TableListingReport generates a summary report all of the tables in
 * a repository.
 */
public class TableListingReport extends AbstractReport
{
    /**
     * Default constructor
     */
    public TableListingReport()
    {
    }

    /**
     * Gets the report name
     * @return the name of the report
     * @see com.sas.plugins.PluginInterface#getName()
     */
    public String getName()
    {
        return "Tables Report";
    } //end method

    /**
     * Gets the report description
     * @return the description of the report
     * @see com.sas.plugins.PluginInterface#getDescription()
     */
    public String getDescription()
    {
        return "Shows a list of all the tables in the repository";
    } //end method

    /**
     * Gets the category that the report will be using. Cannot have
     * multiple levels, only a single level can be used
     * @return the category name
     * @see com.sas.wadmin.plugins.ReportingInterface#getCategory()
     */
    public String getCategory()
    {
        return "Table";
    } //end method

    /**
     * Should return the fully qualified class name of this class. This is
     * used to tie the report to the report visual.
     * Example: com.sas.reports.TableReport
     * @return the report plug-in class name
     * @see com.sas.wadmin.plugins.ReportingInterface#getReportingClass()
     */
    public String getReportingClass()
    {

```



```

        return "com.sas.reports.TableListingReport";
    } //end method

/**
 * This method is called after the user selects a report and exits out
 * of the reporting framework. This method executes on the UI thread
 * and is called prior to when getSourceCode() is sent to the application server.
 * @see com.sas.wadmin.plugins.ReportingInterface#onSelected()
 */
public void onSelected()
{
    //This report doesn't have any extra UI elements
} //end method

/**
 * Gets the report's generated code. This code will then be executed on the
 * application server on a background thread.
 * @return string buffer containing the code to send to the application server
 * @see com.sas.wadmin.plugins.ReportingInterface#getSourceCode()
 */
public StringBuffer getSourceCode()
{
    SASCodeGeneration codeGen = new SASCodeGeneration();

    codeGen.addCommentLine( "Creates an overview or summary report of
                           all tables in the server." );

    WsServerRequest svrRequest = ReportingController.getInstance().
        getServerRequest();
    WsAppServer appServer = svrRequest.getAppServer();

    if (appServer == null)
        return new StringBuffer();

    try
    {
        codeGen.genMetadataMacrosAndOptions( appServer.getServerContext(),
                                           appServer.getServerContext(),
                                           true );
    }
    catch (MdException me)
    {
        MessageUtil.displayMetadataExceptionMessage( me, MessageUtil.
                                                    ACCESSING );
    }
    catch ( java.rmi.RemoteException re )
    {
        com.sas.workspace.Workspace.handleRemoteException( re );
    }

    codeGen.addSourceCode( "filename request temp;\n\n" );
    codeGen.addSourceCode( "data _null_;\n" );
    codeGen.indent( 3 );
    codeGen.addSourceCode( "file request;\n" );
    codeGen.addSourceCode( "infile cards4;\n" );
    codeGen.addSourceCode( "length long $256;\n" );
    codeGen.addSourceCode( "input;\n" );

```

```

codeGen.addSourceCode( "long=_infile_;\n" );
codeGen.addSourceCode( "put long \' \';\n" );
codeGen.unIndent( 3 );
codeGen.addSourceCode( "cards4;\n" );
codeGen.addSourceCode( "<GetMetadataObjects>\n" );
codeGen.addSourceCode( "<ReposId>$METAREPOSITORY</ReposId>\n" );
codeGen.addSourceCode( "<Type>PhysicalTable</Type>\n" );
codeGen.addSourceCode( "<Objects>\n" );
codeGen.addSourceCode( "<ns>SAS</ns>\n" );
codeGen.addSourceCode( "<Flags>260</Flags>\n" );
codeGen.addSourceCode( "<Options>\n" );
codeGen.addSourceCode( "<Templates>\n" );
codeGen.addSourceCode( "<PhysicalTable Name=\"\" Desc=\"\"
                        ChangeState=\"\" MetadataCreated=\"\"
                        MetadataUpdated=\"\">\n" );
codeGen.addSourceCode( "<Trees/> <ResponsibleParties/><TablePackage/>
                        </PhysicalTable>\n" );
codeGen.addSourceCode( "<ResponsibleParty Name=\"\"/>\n" );
codeGen.addSourceCode( "<SASLibrary Name=\"\"/>\n" );
codeGen.addSourceCode( "<Tree Name=\"\"/>\n" );
codeGen.addSourceCode( "<DatabaseSchema Name=\"\"/>\n" );
codeGen.addSourceCode( "</Templates>\n" );
codeGen.addSourceCode( "</Options>\n" );
codeGen.addSourceCode( "</GetMetadataObjects>\n" );
codeGen.addSourceCode( ";;;\n" );
codeGen.addSourceCode( "run;\n\n" );
codeGen.addCommentLine( "Issue the request." );
codeGen.addSourceCode( "filename response temp lrecl=1024;\n\n" );
codeGen.addSourceCode( "proc metadata in=request out=response;\n" );
codeGen.addSourceCode( "run;\n\n" );
codeGen.addCommentLine( "Build the XML Map file to parse the
                        response." );
codeGen.addSourceCode( "filename map temp;\n\n" );
codeGen.addSourceCode( "data _null_;\n" );
codeGen.indent( 3 );
codeGen.addSourceCode( "file map;\n" );
codeGen.addSourceCode( "put \'<?xml version=\"1.0\" ?>\';\n" );
codeGen.addSourceCode( "put \'<SXLEMAP version=\"1.2\">\';\n" );
codeGen.addSourceCode( "put \'<TABLE name=\"Tables\">\';\n" );
codeGen.addSourceCode( "put \'<TABLE-PATH syntax=\"xpath\"/>
                        GetMetadataObjects/Objects/PhysicalTable
                        </TABLE-PATH>\';\n" );
codeGen.addSourceCode( "put \'<COLUMN name=\"table\" retain=\"YES\">
                        \';\n" );
codeGen.addSourceCode( "put \"<PATH>/GetMetadataObjects/Objects/
                        PhysicalTable@Name</PATH>\";\n" );
codeGen.addSourceCode( "put \'<TYPE>character</TYPE>\';\n" );
codeGen.addSourceCode( "put \'<DATATYPE>STRING</DATATYPE>\';\n" );
codeGen.addSourceCode( "put \'<LENGTH>60</LENGTH>\';\n" );
codeGen.addSourceCode( "put \'</COLUMN>\';\n" );
codeGen.addSourceCode( "put \'<COLUMN name=\"description\" retain=
                        \"YES\">\';\n" );
codeGen.addSourceCode( "put \"<PATH>/GetMetadataObjects/Objects/
                        PhysicalTable@Desc</PATH>\";\n" );
codeGen.addSourceCode( "put \'<TYPE>character</TYPE>\';\n" );
codeGen.addSourceCode( "put \'<DATATYPE>STRING</DATATYPE>\';\n" );

```

```

codeGen.addSourceCode( "put \'<LENGTH>200</LENGTH>\' ;\n" );
codeGen.addSourceCode( "put \'</COLUMN>\' ;\n" );
codeGen.addSourceCode( "put \'<COLUMN name=\"created\"
                        retain=\"YES\">\' ;\n" );
codeGen.addSourceCode( "put \'<PATH>/GetMetadataObjects/Objects/
                        PhysicalTable@MetadataCreated</PATH>\' ;\n" );
codeGen.addSourceCode( "put \'<TYPE>date</TYPE>\' ;\n" );
codeGen.addSourceCode( "put \'<DATATYPE>TIME</DATATYPE>\' ;\n" );
codeGen.addSourceCode( "put \'<LENGTH>20</LENGTH>\' ;\n" );
codeGen.addSourceCode( "put \'</COLUMN>\' ;\n" );
codeGen.addSourceCode( "put \'<COLUMN name=\"modified\"
                        retain=\"YES\">\' ;\n" );
codeGen.addSourceCode( "put \'<PATH>/GetMetadataObjects/Objects/
                        PhysicalTable@MetadataUpdated</PATH>\' ;\n" );
codeGen.addSourceCode( "put \'<TYPE>date</TYPE>\' ;\n" );
codeGen.addSourceCode( "put \'<DATATYPE>TIME</DATATYPE>\' ;\n" );
codeGen.addSourceCode( "put \'<LENGTH>20</LENGTH>\' ;\n" );
codeGen.addSourceCode( "put \'</COLUMN>\' ;\n" );
codeGen.addSourceCode( "put \'<COLUMN name=\"owner\" retain=\"YES\">
                        \' ;\n" );
codeGen.addSourceCode( "put \'<PATH>/GetMetadataObjects/Objects/
                        PhysicalTable/ResponsibleParties/
                        ResponsibleParty@Name</PATH>\' ;\n" );
codeGen.addSourceCode( "put \'<TYPE>character</TYPE>\' ;\n" );
codeGen.addSourceCode( "put \'<DATATYPE>STRING</DATATYPE>\' ;\n" );
codeGen.addSourceCode( "put \'<LENGTH>60</LENGTH>\' ;\n" );
codeGen.addSourceCode( "put \'</COLUMN>\' ;\n" );
codeGen.addSourceCode( "put \'<COLUMN name=\"schema\"
                        retain=\"YES\">\' ;\n" );
codeGen.addSourceCode( "put \'<PATH>/GetMetadataObjects/Objects/
                        PhysicalTable/TablePackage/
                        DatabaseSchema@Name</PATH>\' ;\n" );
codeGen.addSourceCode( "put \'<TYPE>character</TYPE>\' ;\n" );
codeGen.addSourceCode( "put \'<DATATYPE>STRING</DATATYPE>\' ;\n" );
codeGen.addSourceCode( "put \'<LENGTH>60</LENGTH>\' ;\n" );
codeGen.addSourceCode( "put \'</COLUMN>\' ;\n" );
codeGen.addSourceCode( "put \'<COLUMN name=\"group\"
                        retain=\"YES\">\' ;\n" );
codeGen.addSourceCode( "put \'<PATH>/GetMetadataObjects/Objects/
                        PhysicalTable/Trees/Tree@Name</PATH>\' ;\n" );
codeGen.addSourceCode( "put \'<TYPE>character</TYPE>\' ;\n" );
codeGen.addSourceCode( "put \'<DATATYPE>STRING</DATATYPE>\' ;\n" );
codeGen.addSourceCode( "put \'<LENGTH>60</LENGTH>\' ;\n" );
codeGen.addSourceCode( "put \'</COLUMN>\' ;\n" );
codeGen.addSourceCode( "put \'<COLUMN name=\"checkout\"
                        retain=\"YES\">\' ;\n" );
codeGen.addSourceCode( "put \'<PATH>/GetMetadataObjects/Objects/
                        PhysicalTable@ChangeState</PATH>\' ;\n" );
codeGen.addSourceCode( "put \'<TYPE>character</TYPE>\' ;\n" );
codeGen.addSourceCode( "put \'<DATATYPE>STRING</DATATYPE>\' ;\n" );
codeGen.addSourceCode( "put \'<LENGTH>60</LENGTH>\' ;\n" );
codeGen.addSourceCode( "put \'</COLUMN>\' ;\n" );
codeGen.addSourceCode( "put \'</TABLE>\' ;\n" );
codeGen.addSourceCode( "put \'</SXLEMAP>\' ;\n" );
codeGen.unIndent( 3 );
codeGen.addSourceCode( "run;\n\n" );

```

```

        codeGen.addCommentLine( "Parse the response with the XML library
                                engine and PROC SQL." );
        codeGen.addSourceCode( "libname response xml xmlmap=map;\n\n" );
        codeGen.addCommentLine( "Create a HTML report for viewing
                                the table." );
        codeGen.addSourceCode( "filename myReport \"")
            .addSourceCode( getURL() )
            .addSourceCode( "\";\n" );
        String sformat = getODSFormatType();
        codeGen.addSourceCode( "ods " )
            .addSourceCode( sformat )
            .addSourceCode( " file=myReport\n" );
        //Check to see if style sheet is being used
        String sStyleSheet = getODSStyleSheet();
        //This options only works with HTML...
        if ( sformat.equals( ReportingInterface.ODS_HTML ))
            if ( sStyleSheet != null && sStyleSheet.length() > 0 )
            {
                codeGen.addSourceCode( "stylesheet=(URL=\"file:" )
                    .addSourceCode( sStyleSheet.trim() )
                    .addSourceCode( "\" )\n" );
            }
        String sAdditionalOptions = getODSAdditionalOptions();
        if ( sAdditionalOptions != null && sAdditionalOptions.length() > 0 )
        {
            codeGen.addSourceCode( sAdditionalOptions )
                .addSourceCode( "\n" );
        }
        codeGen.addSourceCode( ";\n" );

        //Set up the Table column name display
        codeGen.addSourceCode( "%let etls_table = %str(\"" +
        codeGen.escapeMacroValue( "Table Name" ) + "\" );\n" )
            .addSourceCode( "%let etls_descr = %str(\"" +
        codeGen.escapeMacroValue( "Description" ) + "\" );\n" )
            .addSourceCode( "%let etls_create = %str(\"" +
        codeGen.escapeMacroValue( "Created" ) + "\" );\n" )
            .addSourceCode( "%let etls_modified = %str(\"" +
        codeGen.escapeMacroValue( "Last Modified" ) + "\" );\n" )
            .addSourceCode( "%let etls_owner = %str(\"" +
        codeGen.escapeMacroValue( "Owner" ) + "\" );\n" )
            .addSourceCode( "%let etls_schema = %str(\"" +
        codeGen.escapeMacroValue( "Schema" ) + "\" );\n" )
            .addSourceCode( "%let etls_group = %str(\"" +
        codeGen.escapeMacroValue( "Folder" ) + "\" );\n" )
            .addSourceCode( "%let etls_checkout = %str(\"" +
        codeGen.escapeMacroValue( "Checked Out" ) + "\" );\n" );

        codeGen.addSourceCode( "title \"" )
            .addSourceCode( getName() )
            .addSourceCode( "\";\n\n" );
        codeGen.addSourceCode( "proc print data=response.tables label;\n" )
            .addSourceCode( "var table description created modified owner
                            schema group checkout;\n" )
            .addSourceCode( "label table = &etls_table\n" )
            .addSourceCode( "        description = &etls_descr\n" );

```

```

        .addSourceCode( "        created = &etls_create\n" )
        .addSourceCode( "        modified = &etls_modified\n" )
        .addSourceCode( "        owner = &etls_owner\n" )
        .addSourceCode( "        schema = &etls_schema\n" )
        .addSourceCode( "        group = &etls_group\n" )
        .addSourceCode( "        checkout = &etls_checkout\n" );
    codeGen.addSourceCode( "run;\n\n" );
    codeGen.addSourceCode( "ods " )
        .addSourceCode( getODSFormatType() )
        .addSourceCode( " close;\n\n" );
    codeGen.addCommentLine( "Cleanup" );
    codeGen.addSourceCode( "filename request;\n" );
    codeGen.addSourceCode( "filename response;\n" );
    codeGen.addSourceCode( "filename map;\n" );

    return codeGen.getSourceBuffer();
} //end method

} //end class

```

---

## Reporting Interface Methods

New report plug-ins need to implement `com.sas.wadmin.plugins.ReportingInterface`, which is an extension of the `com.sas.plugins.PluginInterface`. Implementation of each of the methods in the Reporting Interface allows the report designer to have control over the Reports window. The `onSelected()` method can be used to generate a report by using Java classes or display dialog boxes to gather additional information needed for the generated source to run. The `getSourceCode()` method returns the SAS code that is submitted to the SAS application server, or it returns null if no code is being used to generate the report.

To add a report to the Analysis window for tables the category needs to be Table Analysis. When running in the Analysis window, the reporting framework supplies the selected default table to your report. When the report is run in the Reports window you need to supply a table. You can add code to the `onSelected()` method to check if the default metadata object is null. If it is, then you can display a dialog box that allows the user to select the table. If you want to add a report to the external table Analysis window, then the category needs to be External Table Analysis.

An abstract implementation of the reporting interface has been provided called `com.sas.wadmin.reports.AbstractReport`. `AbstractReport` provides some default implementations of the interface methods. It assumes that the report is being generated with SAS ODS, and the Output Delivery System Report Options dialog box is being used. The following table shows the main interface methods, explains how they work with the Reports window, and gives a short description of how `AbstractReport` has implemented the methods. For an example of how these methods can be used to create a report, see [“Example Java Code for a Report Plug-in” on page 427](#).

The following table contains information about the methods you can use to create your own report.

**Table A2.1** Reporting Interface Methods

Name	Description	Default Abstract Report Implementation
<code>getName();</code>	This method returns the report name and is displayed in the Name column of the Reports window.	Not implemented.
<code>getDescription();</code>	This method returns the report description and is displayed in the Description column of the Reports window.	Not implemented.
<code>String getCategory();</code>	This method returns the category that the report uses. The report category is displayed in the Reports window under the heading Type. The user can also choose to show reports based on the category name.	Not implemented.
<code>StringBuffer getSourceCode();</code>	This method returns the generated SAS code that can be used to generate a report. Code returned by this method is submitted to the application server by the report framework. If code is not being used to generate the report, then this method returns null.	Not implemented.
<code>void onSelected();</code>	This method executes when the user runs the selected report before submitting any SAS code to the application server that is returned by the <code>getSourceCode()</code> method. This method can be left empty for reports that do not contain any visual elements, or whose processing is done solely with generated SAS code.	Not implemented.

Name	Description	Default Abstract Report Implementation
void setDefaultMetadataObject(Root object);	An optional metadata object might be used when generating the report. This option only is set automatically by the report framework if the report category is set to “Table Analysis” or “External Table Analysis”. These reports are shown in the Analysis window, and the selected table or external table is used as the default object.	Sets a member variable.
Root getDefaultMetadataObject();	This method returns the optional metadata object or null.	Returns the member variable value that is set in the setDefaultMetadataObject method.
void setPath(String path);	The report designer can set a default path or allow the user to set the path in the Reports window. The report framework saves up to eight paths per client in the application default files. These paths are loaded at initialization and set to the first path displayed in the combo box.	Sets a member variable .
String getPath();	This method returns the default path if it is set by the report designer. This method is used by the Reports window to show that default path.	Returns the member variable value that is set in the setPath method.
Boolean isLocalBrowse();	This method returns true if the <b>Browse</b> button on the Report results pane in the Reports window brings up the local file system browser. This method returns false if the remote file system browser is used. This determination is based on the application server dialog box.	Returns false, so the remote file system browser is displayed based on the default application server selected.
void setFileName(String filename);	The report designer can choose to set the default filename to use when the report is created and saved.	Sets a member variable.

Name	Description	Default Abstract Report Implementation
String getFileName();	This method returns the default filename. This method is used by the Reports window to show the default filename.	If the member variable in the setFileName method has not been set, then the returned name is the report name with all spaces removed and the type of ODS format selected (that is, html, .rtf, or .pdf).
Boolean isFileNameFieldEditable();	This method returns false if the user cannot change the filename. The report designer can turn off the user's ability to edit the filename.	Returns true.
String getURL();	This method returns the report URL.	Returns the fully qualified path and filename.
Boolean hasAnOptionsDialog();	This method returns true if the <b>Additional report options</b> button on the Reports window toolbar is activated.	Returns true.
void showOptionsDialog(ReportingController controller)	This method is executed when the user opens the Report Options dialog box for the selected report.	Shows the Report Options dialog box.
void setODSFormatType(String type);	This method is used only if the report is generated with SAS ODS and is using the default Report Options dialog box. This method is called when the user selects HTML, RTF, or PDF in the <b>Format</b> field on the Report Options dialog box. The default is set to HTML.	Sets a member variable.
String getODSFormatType();	This method returns the ODS format type that is selected by the user.	Returns the member variable that is set in the setODSFormatType method. If there is no format type set, then this method returns 'HTML' as the default.
void setODSStyleSheet(String cssFile);	This method is used only if the report is generated with SAS ODS and is using the default Report Options dialog box. This method is called if the user selects an ODS style sheet to be used with the report in the <b>Style</b> field on the Report Options dialog box.	Sets a member variable.



Name	Description	Default Abstract Report Implementation
String getODSStyleSheet();	This method returns the ODS style sheet that is selected by the user.	Returns the member variable that is set in the setODSStyleSheet method.
void setODSAdditionalOptions(String addOptions);	This method is called by the <b>Additional options</b> field in the default Report Options dialog box.	Sets a member variable.
String getODSAdditionalOptions();	This method returns any additional ODS options that are set by the user to be used when generating the report.	Returns the member variable that is set in the setODSAdditionalOptions method.
String getReportingClass();	This method returns the fully qualified class name. This method is called to tie the report to the Reports window. One example is: com.sas.reports.TableReport.	Not implemented.



# 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 also 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.

**CDC**

See change data capture.

**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 data capture**

the process of capturing changes that are made to data, and making these changes available in a machine-readable format. By capturing only the changes in the data, CDC reduces the volume of information that is required for data integration. Short form: CDC.

**change management**

in the SAS Open Metadata Architecture, a facility for metadata source control, metadata promotion, and metadata replication.

**channel**

a virtual communication path for distributing information. In SAS, a channel is identified with a particular topic (just as a television channel is identified with a particular radio frequency). Using the features of the Publishing Framework, authorized users or applications can publish digital content to the channel, and authorized users and applications can subscribe to the channel in order to receive the content. See also publish and subscribe.

**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**

an optional metadata store for a SAS Metadata Server that can be configured in addition to the foundation repository. Custom repositories are useful for physically segregating metadata for storage or security purposes.

**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, and 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.

**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. See also e-mail, message queue, WebDAV (Web Distributed Authoring and Versioning), and channel.

**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.

**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.

**e-mail**

a system for transmitting messages electronically, usually between two computers. See also delivery transport.

**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**

a column or combination of columns in one table that references the corresponding primary key in another table. A foreign key must have the same attributes as the primary key that it references.

**foundation repository**

the required metadata store for a SAS Metadata Server. Each SAS Metadata Server has one foundation repository that is created by default when the metadata server is configured.

**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 and data lineage.

**Integrated Object Model server**

a SAS object server that is launched in order to fulfill client requests for IOM services. Short form: IOM server.

**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.

**IOM server**

See Integrated Object Model server.

**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 English\_USA.

**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.

**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.

**PFD**

See process flow diagram.

**primary key**

a column or combination of columns that uniquely identifies a row in a table.

**process flow diagram**

a 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. Each process in the diagram is specified by a metadata object called a transformation. Short form: PFD.

**project repository**

a metadata repository that serves as an individual work area or playpen. Project repositories are available for SAS Data Integration Studio only. In general, each user who participates in change management has his or her own project repository.

**publish**

to deliver electronic information, such as SAS files (including SAS data sets, SAS catalogs, and SAS data views), other digital content, and system-generated events to one or more destinations. These destinations can include e-mail addresses, message queues, publication channels and subscribers, WebDAV-compliant servers, and archive locations.

**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 Stored Process Server**

a SAS IOM server that is launched in order to fulfill client requests for SAS Stored Processes. See also IOM server.

**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 SAS libraries.

**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 amount of information in match codes. Greater sensitivity values result in match codes that contain greater amounts of information. As sensitivity values increase, character values must be increasingly similar to generate the same match codes.

**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.

**subscribe**

to sign up to receive electronic content that is published to a SAS publication channel.

**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.

**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.

**Web Distributed Authoring and Versioning**

an emerging industry standard, based on extensions to HTTP 1.1, that enables users to collaborate in the development of files and collections of files on remote Web servers. Short form: WebDAV. See also delivery transport.

**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.

**WebDAV**

See Web Distributed Authoring and Versioning.



# Index

---

## A

Access folder [414](#)  
 accessibility features [10](#)  
 actions  
   based on job status [173](#)  
   based on transformation status [175](#)  
   default [168](#)  
   prerequisites for [172](#)  
 administrative tasks [9](#)  
 aggregate columns [237](#)  
 Analysis folder [414](#)  
 Analysis window [396](#)  
 application servers  
   modifying configuration files or SAS  
     start commands for [211](#)  
 Archived folder [415](#)  
 assistive technologies [10](#)  
 Authorization tab [23](#)  
 automatic column mappings [154](#)  
 automatic joins [291](#)  
 automatic propagation [158](#)

## B

Basic Properties pane [408](#)  
 browsing table data [89](#)  
 buffering options [330](#)  
 bulk load tables [329](#)  
 bulk loading [277](#)

## C

case  
   in table and column names [72](#)  
 CASE expressions [305](#)  
 change data capture (CDC) [369](#)  
   changed data tables [371](#)

  control tables [372](#)  
   from Oracle [372](#)  
   prerequisites for [370](#)  
 Change Data Capture folder [415](#)  
 Change Data Capture transformations  
   [369](#)  
 change detection  
   dimension tables [350](#)  
 change management [36](#)  
   adding metadata [38](#)  
   checking in metadata [38](#)  
   checking out metadata [39](#)  
   clearing metadata from projects [40](#)  
   creating connection profiles for  
     administrators [38](#)  
   creating connection profiles for users  
     [37](#)  
   deleting metadata [39](#)  
   migration and [17](#)  
   undoing checkouts [40](#)  
   usage notes for [40](#)  
 change tracking  
   datetime [365](#), [367](#)  
   dimension tables [350](#)  
 checkouts  
   undoing [40](#)  
 Checkouts tree [397](#)  
 clauses  
   adding subqueries to [317](#)  
   reviewing and modifying [288](#)  
 cleansing data [4](#), [235](#)  
 clearing metadata [40](#)  
 cluster tables [389](#)  
 COBOL copybooks [115](#)  
 COBOL data files [115](#)  
 code  
   common code generated for jobs [138](#)

- credentials in generated code 140
    - displaying SAS code for jobs 137
    - for transformations 164
    - generated 205
    - jobs with generated source code 122
    - jobs with user-supplied source code 123
    - overriding generated code 112
    - user-written 213
    - user-written SQL code 330
  - Code Editor 397
  - code testing 108
  - column heading options 97
  - column mappings 154
    - automatic 154
    - deleting 157
    - derived 156
    - from source table to work table 218
    - from work table to target table 218
    - one-to-one 156
    - options for 157
  - column metadata
    - adding 77
    - additional operations 80
    - maintaining 77
    - modifying 78
    - notes and documents for 79
  - column names
    - case and special characters in 72
  - columns
    - adding to query target table 299
    - aggregate columns 237
    - avoiding unneeded columns 236
    - deleting from indexes 88
    - dropping unneeded columns 236
    - key columns 82
    - managing for performance 235
    - matching variable size to data length 237
    - rearranging in indexes 88
    - scope of column changes in jobs 158
    - updating columns in keys 86
  - Comparison Results window 398
  - conditional action sets
    - default 168
  - conditions
    - default 168
  - configuration files
    - modifying for application servers 211
  - Connection Profile window 399
  - connection profiles
    - creating 26
    - creating for administrators, with change management 38
    - creating for users, with change management 37
    - opening 26
    - updating 27
  - connections for objects 162
  - connectivity 3
  - constraints
    - removing non-essential constraints during a load 276
  - Control Flow tab 149
  - Control folder 416
  - control tables 340
    - registering 340
  - credentials
    - in generated code 140, 208
  - cross-reference tables 352
  - cube wizards 424
  - cubes
    - registering 33
- D**
- data cleansing 4, 235
  - data enrichment 4
  - data federation 4
  - Data folder 416
  - data integration 3
    - advantages of 4
  - data integration environment 5
    - libraries 8
    - overview diagram 5
    - SAS Data Integration Studio 6
    - SAS Management Console 5
    - servers 6
  - data optimization 323
  - Data Quality folder 417
  - data surveyor wizards 425
  - Data Transfer transformation 135
  - Data Transforms folder 418
  - data validation 235
  - datetime change tracking 365
    - closing out rows in 367
  - DBMS names 74
    - setting options in Register Tables wizard 76
  - DBMS servers 7
  - DBMS tables
    - preserving foreign keys in 35
  - debugging 238
    - adding code to process flows 240
    - checking job status 239
    - limiting input to transformations 239
    - setting and checking status codes 241
    - setting SAS invocation options on jobs 241
  - SQL queries 297
  - verifying output from transformations 239

- default actions 169
- default conditional action sets 171
- default conditions 168
- default SAS Application Server 30
- Delimited External File wizard 101
- delimited external files
  - registering 100
- deploying jobs 184
  - as stored processes 189, 190
  - as Web services 194
  - creating Web service jobs 196
  - for execution on remote host 188
  - for scheduling 185
  - prerequisites for deploying as stored process 190
  - prerequisites for scheduling 185
  - prerequisites for Web service jobs 195
  - redeploying jobs for scheduling 187
  - redeploying to stored processes 192
  - requirements for Web service jobs 195
  - scheduling for complex process flows 187
  - stored process as Web service 203
  - viewing or updating stored process metadata 193
  - Web service jobs as stored process 200
- derived column mappings 156
- Designer tab (SQL)
  - adding CASE expression 306
  - adding GROUP BY clause 310
  - adding HAVING clause 310
  - adding joins to queries 299
  - adding ORDER BY clause 312
  - submitting queries from 318
- Designer window 287
- desktop 399
- Details pane 401
- Diagram tab 150
- dimension table lookup 356
- dimension tables 350
  - change detection and loading for 350
  - change tracking 350
  - cross-reference tables and 352
  - generated keys and 351
  - loading with Type 1 and 2 updates 353
  - Type 1 updates 352
- document objects
  - saving reports as 262
- documentation 9
  - for process flow diagrams 133
- documents
  - adding to registered objects 41
  - for columns 79
  - viewing contents of 43

## E

- editing options 98
- empty jobs 123
- encoding options 113
- environment 5
- ERM systems 8
- error log location 25
- ETL (extraction, transformation, and loading) 4
- executing jobs
  - on remote host 188
- explicit pass-through processing 327
- exporting metadata 46
  - documenting 48
  - SAS Metadata Bridges 51, 60
  - SAS Package metadata 46, 47, 48
  - selected metadata 48
- Expression Builder window 402
- external files 100
  - accessing with FTP or HTTP server 113
  - browse and edit options for 96
  - common tasks 100
  - in job process flows 117
  - NLS support for 113
  - overriding code generated by wizards 112
  - registering COBOL data files 115
  - registering delimited files 100
  - registering files with user-written code 108
  - registering fixed-width files 103
  - updating metadata 111
  - viewing data in 114
  - viewing metadata 111
- extraction, transformation, and loading (ETL) 4

## F

- fact tables 352
  - loading with dimension table lookup 356
  - structure and loading of 352
- fixed-width external file wizard 104
- fixed-width external files
  - registering 103
- Folders tree 27, 403
  - adding folders 28
  - adding metadata objects to folders 29
  - changing folder paths 30
  - icon overlays for metadata objects 406
  - renaming folders 29
- foreign keys
  - adding 85
  - applying changes to tables 59

- key columns 82
- preserving in DBMS tables 35
- restoring metadata for 60
- format libraries
  - specifying in preprocess to job 32
- format options 97
- formats
  - user-defined 32
- FTP servers
  - accessing external files 113

**G**

- generated code 205
  - credentials in 140, 208
  - displaying for jobs 209
  - displaying for transformations 209
  - editing for jobs 228
  - for jobs 138
  - jobs with generated source code 122
  - LIBNAME statements and 206
  - macro variables and 207
  - macro variables for status handling in 177
  - modifying configuration files for
    - application servers 211
  - modifying SAS start commands 211
  - options for jobs 210
  - options for transformations 210
  - overriding, when created by External File wizards 112
  - remote connection statements and 207
  - replacing for jobs 229
  - SYSLAST macro statement and 206
- generated keys 351
- generated transformations 198, 219
  - impact analysis on 226, 251
  - maintaining 226
  - updating 227
- global options 35
  - for jobs 131
  - for tables 70
- global Options window 35
- grids
  - submitting jobs to 144
- GROUP BY clause
  - adding to queries 309

**H**

- hash joins 325
- HAVING clause
  - adding to queries 309
- Help 8
- hiding logs 243
- HTTP servers

- accessing external files 113

**I**

- I/O processing
  - minimizing 323
- impact analysis 247
  - on generated transformations 226
  - performing 248
  - performing on generated transformations 251
  - reverse 247, 253
- implicit pass-through processing 327
- implicit property for joins 325
- Import and Export Metadata wizards 425
- importing COBOL copybooks 116
- importing metadata 46
  - comparing metadata to repository 56
  - invalid change analysis result 60
  - SAS Metadata Bridges 51
  - SAS Package metadata 46, 47, 49
- importing SAS metadata
  - comparing metadata to repository 58
- index joins 324
- indexes 87
  - creating 87
  - deleting 88
  - deleting columns from 88
  - rearranging columns in 88
  - removing non-essential indexes during a load 276
- INFILE statement
  - replacing generated statements 112
- input tables
  - adding subqueries to 314
- intermediate files
  - deleting for performance 233
- invalid change analysis results 60
- Inventory tree 404
  - icon overlays for metadata objects 406
- invocation options
  - setting on jobs 241
- iterative jobs 333
  - adding input and transformation directly 335
  - creating and running 334
  - creating control tables 340
  - creating parameterized jobs 337

**J**

- Java code
  - for report plug-ins 427
- Java options 24
- Job Documentation Report
  - customizing 259



- Job Editor window 407
    - making connections in 162
    - submitting jobs from 143
    - viewing code in 209
  - job management 142
  - job metadata
    - viewing or updating 136
  - job options 130, 210
    - global 210
    - local 210
  - job properties
    - viewing or updating 136
  - job properties window 409
  - job statistics
    - prerequisites for collecting 145
  - job status 140, 239
    - actions based on 173
  - jobs 122
    - accessing local and remote data 133
    - adding User Written Code transformation to 216
    - column mappings 154
    - common code generated for 138
    - creating empty jobs 123
    - creating jobs containing jobs 125
    - creating process flows for 124
    - credentials in generated code 140
    - data access in context of 134
    - Data Transfer transformation 135
    - default temporary output tables 126
    - deploying 184
    - displaying generated code for 209
    - displaying SAS code for 137
    - editing generated code for 228
    - external files in process flows 117
    - generated code for 205
    - global options for 131
    - interactive data access 134
    - iterative 333
    - local options for 132
    - managing submitted jobs 123
    - parameterized 320, 333, 337
    - replacing generated code for 229
    - reviewing successful jobs 145
    - running 123
    - SAS invocation options on 241
    - scope of column changes 158
    - status handling for 167
    - submitting for immediate execution 142
    - submitting one step at a time 144
    - submitting queries from 318
    - submitting segments of 144
    - submitting selected transformations 143
    - submitting to a grid 144
    - troubleshooting 150
    - updates during partial promotion 15
    - updating during migration 14
    - user action after migration 15
    - viewing or updating metadata 136
    - with generated source code 122
    - with user-supplied source code 123
  - join algorithms 324
  - join types
    - changing 294
    - results by 295
  - joins 287
    - adding to queries on Designer tab 299
    - automatic 291
    - hash joins 325
    - implicit property for 325
    - index joins 324
    - joining a table to itself 319
    - parameters with 320
    - reviewing and modifying 288
    - selecting join type 294
    - sort-merge 324
    - SPD Server star joins 321
- K**
- key columns 82
  - Key Effective Date transformation 349
  - keys
    - adding primary or unique keys 85
    - deleting 87
    - generated 351
    - maintaining 82
    - renaming 87
    - surrogate 238
    - surrogate primary 362
    - updating columns in 86
    - viewing 83
- L**
- level\_value option 25
  - LIBNAME statement
    - generated code and 206
    - generated job code and 138
  - libraries 8
    - registering 31
  - List Cluster Contents transformation 391
  - List Data transformation
    - adding to process flows 246
  - load techniques 273
    - adding rows 275
    - matching and updating rows 275
    - removing all rows 274
  - loader transformations 269
    - bulk loading 277

- removing non-essential indexes and constraints during a load 276
- selecting a load technique 273
- SPD Server Table Loader transformation 270
- Table Loader options 271
- Table Loader transformation 270
- loading output tables 269
- local data
  - accessing 133
- local options
  - for jobs 132
  - for tables 70
- Log tab 152
- logs
  - capturing additional options 242
  - error log location 25
  - evaluating 242
  - for process flow optimization 242
  - hiding 243
  - message logging 25
  - redirecting large logs to a file 243
  - viewing 243
- Lookup transformation 349
- lookups
  - transformations for 238

## M

- macro variables
  - for status handling 140, 177
  - for status handling in generated code 177
  - for status handling in user-written code 182
- generated code and 207
- manual propagation 159
- mappings 154
- master data management 4
- memory allocation 25
- message logging 25
- message queues 379
  - Microsoft queues 386
  - polling WebSphere queues 384
  - prerequisites 380
  - supported data transfer types 379
  - transformations for 381
  - WebSphere queues 382
- metadata
  - adding 38
  - checking in 38
  - checking out 39
  - clearing from projects 40
  - connectivity and 3
  - deleting 39
  - import and export wizards 425

- importing and exporting 46
- maintaining column metadata 77
- updating external file metadata 111
- updating table metadata 68
- viewing external file metadata 111
- viewing or updating job metadata 136
- viewing or updating stored process metadata 193
- viewing or updating transformations metadata 165
- viewing table metadata 68
- metadata objects
  - adding notes or documents to registered objects 41
  - adding to folders 29
  - copying and pasting 51
  - copying to folders 29
  - dragging to folders 29
  - icon overlays for 406
  - moving to folders 29
- Microsoft message queues 386
- migration 4, 13
  - Web site for 17

## N

- names
  - DBMS names 74
  - SAS names 73
- New Job wizard 123
- new object wizards 423
- New Table wizard 66
  - registering tables with 66
- New User-Written External File wizard 108

## NLS

- encoding options 113
- support for external files 113
- notes
  - adding to registered objects 41
  - for columns 79
  - viewing contents of 43

## O

- objects
  - connections for 162
  - migration of 14
- one-to-one column mappings 156
- online Help 8
- Options window 412
- Oracle
  - capturing changed data from 372
- ORDER BY clause
  - adding to queries 312
- Output folder 418

- output tables
  - loading in process flows 269
  - temporary 126
- P**
- parallel processing 342
  - prerequisites for 343
  - setting options for 344
- parameterized jobs 320, 333
  - creating 337
- pass-through processing 326
- performance
  - data optimization 323
  - process flow optimization 231
  - sort performance 279
  - SQL processing 322, 328
- physical tables
  - managing for performance 232
  - updating table metadata 69
- plug-in location 24
- pre-sorting data 323
- Precode and Postcode tab
  - adding user-written code to 214
- primary keys
  - adding 85
  - key columns 82
  - surrogate 362
- process data management 232
- process flow diagrams
  - documenting 133
  - viewing or updating 136
- process flow optimization 231
  - additional information 246
  - column management 235
  - debugging techniques 238
  - logs for 242
  - process data management 232
  - reviewing temporary output tables 244
  - streamlining process flow components 237
- process flows
  - adding debugging code to 240
  - adding List Data transformation to 246
  - adding User Written Code transformation to 246
  - creating 23
  - creating for jobs 124
  - external files in 117
  - loading output tables in 269
  - scheduling for complex process flows 187
  - streamlining components 237
- projects
  - clearing metadata from 40
- promotion

- updates to jobs and transformations 15
- propagation
  - automatic 158
  - manual 159
  - path options 160
- properties windows 408
- Publish folder 418

- Q**
- queries 287
  - adding a GROUP BY clause 309
  - adding a HAVING clause 309
  - adding an ORDER BY clause 312
  - adding CASE expressions 305
  - adding columns to target table 299
  - adding joins to, on Designer tab 299
  - adding user-written SQL code 296
  - configuring a SELECT clause 303
  - creating or configuring a WHERE clause 307
  - creating simple queries 301
  - debugging 297
  - Designer window 287
  - reviewing and modifying clauses, joins, and tables 288
  - submitting 318
  - submitting as part of a job 318
  - submitting from Designer tab 318
  - subqueries 313
  - validating 318

- R**
- redeploying jobs
  - for scheduling 187
  - to stored processes 192
- redirecting
  - logs 243
  - temporary output tables 245
- Register Tables wizard 65, 424
  - setting name options in 76
- registered objects
  - adding notes or documents to 41
- registering
  - COBOL data files 115
  - control tables 340
  - cubes 33
  - delimited external files 100
  - external files with user-written code 108
  - fixed-width external files 103
  - libraries 31
  - tables 33, 65, 66
- remote connection statements 207
  - generated code and 139

- remote data
  - accessing 133
  - minimizing access 235
- remote host
  - deploying jobs for execution on 188
- Remove Cluster transformation 391
- report plug-ins
  - example Java code for 427
  - reporting interface methods 433
- reporting interface methods 433
- reports 256
  - creating 265
  - customizing the Job Documentation Report 259
  - customizing the Tables Report 258
  - opening 263
  - running and saving 260
  - saving as document object 262
  - viewing 263
- Reports window 256, 411
  - selecting a perspective 257
- reverse impact analysis 247, 253
- rows
  - adding 275
  - closing out in datetime change tracking 367
  - matching and updating 275
  - removing all 274
- running jobs 123

## **S**

- SAS Application Servers 6, 30
- SAS code
  - displaying for jobs 137
- SAS Data Integration Studio
  - documentation 9
  - environment and 6
  - online Help 8
  - required components 22
  - setup 22
  - starting 24
  - upgrading 13
- SAS data servers 7
- SAS Grid Server 6
- SAS Intelligence Platform
  - documentation 9
- SAS invocation options
  - setting on jobs 241
- SAS Management Console
  - environment and 5
- SAS Metadata Bridges 51
  - exporting metadata 60
  - importing new metadata 53
  - importing updated metadata 55
  - preparing to import or export 53

- usage notes for 52
- SAS Metadata Server 6
  - connecting to 26
  - reconnecting to 27
- SAS names 73
- SAS OLAP Server 6
- SAS Package
  - importing and exporting metadata 46, 48, 49
  - preparing to import or export metadata 47
- SAS Package wizards 425
- SAS solutions
  - migration and 17
- SAS start commands
  - modifying for application servers 211
- SAS Workspace Server 6
  - requirements for new jobs 16
- SAS/CONNECT Server 6
- SAS/SHARE Server 7
- SCD
  - See [slowly changing dimensions \(SCD\)](#)
- SCD Type 2 Loader transformation 269, 349
- scheduling
  - deploying jobs for 185
  - for complex process flows 187
  - prerequisites for deploying jobs 185
  - redeploying jobs for 187
  - to execute jobs on remote host 188
- search options 97
- Section 508 10
- security 22
- SELECT clause
  - configuring 303
- SELECT statement
  - optimizing 329
- servers 6
- setup 22
- slowly changing dimensions (SCD) 348
  - closing out rows in datetime change tracking 367
  - dimension tables 350
  - fact tables 352
  - loading dimension tables with Type 1 and 2 updates 353
  - loading fact tables with dimension table lookup 356
  - loading tables and adding surrogate primary key 362
  - project stages 349
  - star schema loading process 349
  - star schemas and 348
  - tracking changes in source datetime values 365
  - transformations supporting 349

- types of 348
  - Sort transformation 279
    - creating tables containing sorted contents of a source 282
  - sort transformations 279
    - optimizing sort performance 279
  - sort-merge joins 324
  - sorting
    - performance and 279, 282
    - pre-sorting data 323
  - source code
    - jobs with generated source code 122
    - jobs with user-supplied source code 123
  - source tables
    - mapping columns to work table 218
  - sources
    - table containing sorted contents of 282
  - SPD Server 7
  - SPD Server cluster tables 389
    - creating 390
    - maintaining 391
  - SPD Server Dynamic Cluster folder 419
  - SPD Server star joins 321
  - SPD Server Table Loader transformation 270
  - special characters
    - in table and column names 72
  - SQL Join transformations 287
    - adding a GROUP BY clause 309
    - adding a HAVING clause 309
    - adding an ORDER BY clause 312
    - adding CASE expressions 305
    - adding columns to target table 299
    - adding joins to queries on Designer tab 299
    - adding subqueries 313
    - adding user-written SQL code 296
    - automatic joins 291
    - configuring a SELECT clause 303
    - creating or configuring a WHERE clause 307
    - creating simple queries 301
    - data optimization 323
    - debugging queries 297
    - Designer window 287
    - implicit property for a join 325
    - join algorithms 324
    - joining a table to itself 319
    - optimizing processing performance 322, 328
    - parameters with joins 320
    - pass-through processing 326
    - selecting join type 294
    - SPD Server star joins 321
    - submitting queries 318
    - validating queries 318
  - SQL Properties window
    - options for optimizing performance 328
  - star joins 321
  - star schemas
    - loading process 349
    - slowly changing dimensions and 348
    - transformations for 238
  - starting SAS Data Integration Studio 24
  - Statistics tab 146
  - status codes
    - setting and checking 241
  - status handling
    - actions based on 173
    - actions based on transformation status 175
    - default conditions, actions, and conditional action sets 168
    - for jobs and transformations 167
    - in generated code 177
    - in user-written code 182
    - macro variables for 140, 177
    - prerequisites for actions 172
  - Status tab 145, 151
  - stored process server 6
  - stored processes
    - deploying as Web service 203
    - deploying jobs as 189, 190
    - deploying Web service jobs as 200
    - prerequisites for 190
    - prerequisites for deploying jobs as 190
    - redeploying jobs to 192
    - viewing or updating metadata 193
  - subqueries 313
    - adding to clauses 317
    - adding to input tables 314
  - Surrogate Key Generator transformation 349
  - surrogate keys 238
  - surrogate primary keys 362
  - synchronization 4
  - SYSLAST macro statement
    - generated code and 206
    - job code and 138
- ## T
- Table Loader transformation 270
    - setting options 271
  - table metadata
    - updating with physical table 69
    - viewing or updating 68
  - table names
    - case and special characters in 72
    - default name options 76
    - name options for registered tables 76

- table properties window 410
  - tables 64
    - browse and edit options for 96
    - browsing data 89
    - bulk load 329
    - CDC control tables 372
    - changed data tables 371
    - common tasks 64
    - containing sorted contents of a source 282
    - control tables 340
    - creating with View Data window 95
    - cross-reference tables 352
    - default temporary output tables 126
    - dimension tables 350, 353
    - editing data 92
    - fact tables 352, 356
    - global options for 70
    - indexes 87
    - joining a table to itself 319
    - key columns 82
    - loading and adding surrogate primary key 362
    - local options for 70
    - managing physical tables for performance 232
    - registering 33
    - registering with New Table wizard 66
    - registering with Register Tables wizards 65
    - reviewing and modifying, in queries 288
    - setting default name options 76
    - source tables 218
    - SPD Server cluster tables 389
    - specifying options for 70
    - target tables 218
    - temporary output tables 244
    - updating metadata with physical table 69
    - work tables 218
  - Tables Report
    - customizing 258
  - target tables
    - adding columns to 299
    - mapping columns from work table 218
  - temporary output tables 126
    - preserving 244
    - redirecting 245
    - reviewing 244
    - viewing 245
  - testing code 108
  - threaded reads 330
  - Tools-Options window 412
  - tracking
    - datetime change tracking 365, 367
  - transformation options 210
  - transformation properties window 409
  - transformations 413
    - actions based on status of 175
    - adding directly to iterative jobs 335
    - Change Data Capture 369
    - Data Transfer 135
    - displaying generated code for 209
    - editing generated code for 228
    - for lookups 238
    - for message queues 381
    - for star schemas 238
    - generated 219, 226, 251
    - Key Effective Date 349
    - limiting input 239
    - List Cluster Contents 391
    - List Data 246
    - loader transformations 269
    - Lookup 349
    - Remove Cluster 391
    - replacing generated code for 229
    - SCD Type 2 Loader 269, 349
    - Sort transformations 279
    - SPD Server Table Loader 270
    - SQL Join 287
    - status handling for 167
    - submitting selected transformations 143
    - supporting slowly changing dimensions 349
    - Surrogate Key Generator 349
    - Table Loader 270
    - updates during partial promotion 15
    - updating during migration 14
    - User Written Code 216, 246
    - verifying output 239
    - viewing code for 164
    - viewing or updating metadata 165
    - Websphere Queue Reader 383, 384
    - Websphere Queue Writer 382, 383
  - Transformations tree 413
  - tree view 419
    - after migration 16
  - troubleshooting unsuccessful jobs 150
  - Type 1 updates 352, 353
  - Type 2 updates 353
- ## U
- undoing checkouts 40
  - unique keys
    - adding 85
    - key columns 82
  - update table metadata feature 69
  - updates
    - Type 1 352, 353
    - Type 2 353

- upgrading [13](#)
- User Written Code transformation
  - adding to process flows [246](#)
- user-defined formats [32](#)
- user-supplied source code [123](#)
- user-written code [213](#)
  - adding to Precode and Postcode tab [214](#)
  - editing generated code for jobs or transformations [228](#)
  - generated transformations and [219](#)
  - macro variables for status handling in [182](#)
  - maintaining generated transformations [226](#)
  - replacing generated code for jobs or transformations [229](#)
- SQL [296](#), [330](#)
- User-Written Code transformation
  - adding to jobs [216](#)
- user-written external files
  - registering [108](#)

## **V**

- validating data [235](#)
- View Data window [421](#)
  - creating tables [95](#)
- View File window [422](#)
- views

- managing for performance [232](#)

## **W**

- Warnings and Errors tab [151](#)
- Web services
  - creating jobs [196](#)
  - deploying jobs as [194](#)
  - deploying jobs as stored process [200](#)
  - deploying stored processes as [203](#)
  - prerequisites for deploying jobs [195](#)
  - requirements for deploying jobs [195](#)
- WebSphere message queues [382](#)
  - polling [384](#)
- Websphere Queue Reader transformation
  - configuring and running jobs [383](#)
  - creating jobs [383](#)
  - verifying jobs [384](#)
- Websphere Queue Writer transformation
  - configuring and running jobs [383](#)
  - creating jobs [382](#)
  - verifying jobs [383](#)
- WHERE clause
  - creating or configuring [307](#)
- wizards [423](#)
- work tables
  - mapping columns from source table [218](#)
  - mapping columns to target table [218](#)





---

## Your Turn

We welcome your feedback.

- If you have comments about this book, please send them to **`yourturn@sas.com`**. Include the full title and page numbers (if applicable).
- If you have comments about the software, please send them to **`suggest@sas.com`**.



# SAS® Publishing Delivers!

Whether you are new to the work force or an experienced professional, you need to distinguish yourself in this rapidly changing and competitive job market. SAS® Publishing provides you with a wide range of resources to help you set yourself apart. Visit us online at [support.sas.com/bookstore](http://support.sas.com/bookstore).

## SAS® Press

Need to learn the basics? Struggling with a programming problem? You'll find the expert answers that you need in example-rich books from SAS Press. Written by experienced SAS professionals from around the world, SAS Press books deliver real-world insights on a broad range of topics for all skill levels.

**[support.sas.com/saspress](http://support.sas.com/saspress)**

## SAS® Documentation

To successfully implement applications using SAS software, companies in every industry and on every continent all turn to the one source for accurate, timely, and reliable information: SAS documentation. We currently produce the following types of reference documentation to improve your work experience:

- Online help that is built into the software.
- Tutorials that are integrated into the product.
- Reference documentation delivered in HTML and PDF – **free** on the Web.
- Hard-copy books.

**[support.sas.com/publishing](http://support.sas.com/publishing)**

## SAS® Publishing News

Subscribe to SAS Publishing News to receive up-to-date information about all new SAS titles, author podcasts, and new Web site features via e-mail. Complete instructions on how to subscribe, as well as access to past issues, are available at our Web site.

**[support.sas.com/spn](http://support.sas.com/spn)**



**THE  
POWER  
TO KNOW®**

